DOCUMENT RESUME

ED 302 182 IR 013 541

AUTHOR Yankelovich, Nicole; And Others
TITLE Issues in Designing a Hypermedia Document System: The Intermedia Case Study.
INSTITUTION Brown Univ., Providence, RI. Inst. for Research in Information and Scholarship.
PUB DATE Jun 86
NOTE 53p.; Paper presented at an Invitational Conference on Multimedia in Education (Cupertino, CA, June 19-20, 1986). For the complete journal, see IR 013 538.
PUB TYPE Journal Articles (080) -- Speeches/Conference Papers (150)
JOURNAL CIT Learning Tomorrow: Journal of the Apple Education Advisory Council; n3 p35-87 Spr 1987

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS Case Studies; Computer Graphics; *Computer Software; Computer System Design; *Design Requirements; Electronic Mail; *Information Processing; *Man Machine Systems; *Programing
IDENTIFIERS *Hypermedia; *Software Development

ABSTRACT

Intermedia, a hypermedia system developed at Brown University's Institute for Research (Rhode Island) in Information and Scholarship, is first described, and then used as a case study to explore a number of key issues that software designers must consider in the development of hypermedia document systems. A hypermedia document system is defined as a system that provides a set of tools which allow authors to share a network of linked documents, link their own and others' documents together, leave notes for one another, and retrieve information stored in documents of different types. The major issues addressed include various design options for linking together multimedia documents, the contexts in which those links exist, and the need for visual representation of the links that exist within a given context. Twenty-one figures illustrate the text. (28 references) (Author/EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED302182

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Issues in Designing a Hypermedia Document System

The Intermedia Case Study

Nicole Yankelovich, Karen E. Smith, L. Nancy Garrett, and Norman Meyrowitz

Institute for Research in Information and Scholarship (IRIS)
Brown University

IR013541

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Sau[illegible]ckman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Abstract

A hypermedia document system provides a set of tools that allow authors to share a network of linked documents, link their own and others' documents together, leave notes for one another, and retrieve information stored in documents of different types. This paper describes the Intermedia system developed at Brown University's Institute for Research in Information and Scholarship and then uses Intermedia as a case study to explore a number of key issues that software designers must consider in the development of hypermedia document systems. The major issues we address include a variety of design options for linking together multimedia documents, the contexts in which those links exist, and the need for visual representations of the links that exist within a given context.

Introduction

Annotating existing work and forging links between writings has long been an important undertaking of knowledge workers. Today, computer-based *hypermedia* systems model traditional handwritten margin notes, annotations in illuminated manuscripts, and conventional "see also" references in encyclopedias [Enge68, Meyr85, Nels81, and Yank85]. Such systems provide an electronic equivalent of the cooperative processes of knowledge workers by allowing authors to create, annotate, and link together information from a variety of media such as text, graphics, timelines, video, audio, and spreadsheets.

Hypermedia systems differ from the traditional methods in several important ways. Where the traditional methods imply sequential access, hypermedia systems aim to allow multiple authors to add commentary to the same corpus at the same time. And where the traditional methods are limited to paper media, hypermedia systems are free to exploit the worlds of electronic information access. Where the traditional methods are associated with static textual documents and manually created indices, hypermedia systems are able to present visualizations of a complex, changing, dynamic world.

The design and implementation of a system with such potential power, flexibility, and wide-ranging audience encompasses many degrees of freedom. For a given audience, what is the best way to organize links? What is the best way to symbolize links? What is the best way to present visual renderings of the entire network of links?

This paper catalogs many issues associated with building a hypermedia system. It draws on our experience in designing and building *Intermedia*, a framework for a collection of tools that allow authors to link together the contents of text, timeline, and graphics documents over a network of high-powered workstations [Meyr85, Meyr86, Garr86, Haan86, Yank86, Yank87]. First we describe Intermedia. Next, to present the basic problem domain, we discuss issues involved in the most elementary hypermedia system, one in which a single user is able to link together read-only documents. We then explore many of the same issues as they are complicated by the ability to edit documents, and examine these issues in a multiple-user environment. We then explain the uses of Intermedia, and finish by discussing some of the problems associated with hypermedia systems.

Description of Intermedia

2.1. Hypertext and Hypermedia

In the early 1960s, Theodor Nelson coined the word *hypertext* to describe the idea of nonsequential writing. A hypertext system is one that allows authors or groups of authors to *link* information together, create *paths* through a corpus of related material, *annotate* existing texts, and create notes that point readers to either bibliographic data or the body of the referenced text. With a computer-based hypertext system, students and researchers are not obliged to search through library stacks to look up referenced books and articles; they can quickly follow trails of footnotes without losing their original context. Explicit connections—links—allow readers to travel from one document to another, automating what one does when following references in an encyclopedia. In addition, hypertext systems that support multiple users allow researchers, professors, and students to communicate and collaborate with one another within the context of a body of scholarly material. For a survey of early hypertext systems refer to Yankelovich, 1985.

Hypermedia is simply an extension of hypertext that incorporates other media in addition to text. With a hypermedia system, authors can create a linked corpus of material that includes text, static graphics, animated graphics, video, sound, music, and so forth. Examples and descriptions of existing hypermedia systems can be found in Backer, 1982, Bender, 1984, Feiner, 1982, Halasz, 1985, and Weyer, 1985.

2.2. Architecture

Intermedia is both an author's tool and a reader's tool. The system, in fact, makes no distinction between types of users providing they have appropriate access rights to the material they wish to edit, explore, or annotate. Creating new materials, making links and following links, are all integrated into a single, modeless environment.

The Intermedia system is built on top of the 4.2 BSD UNIX operating system and runs on IBM RT/PC and Sun workstations that support Sun's Network File System (NFS). To create Intermedia, the software-development team adapted an object-oriented preprocessor to the C programming language licensed to Brown University by Bolt, Beranek, and Newman as well as Apple's MacApp facility for creating generic Macintosh applications [Tesl85] and Cadmus's CadMac tool box, both under special agreement with Apple Computer, Inc. With an object-oriented development environment and a UNIX-based implementation of the Macintosh Toolbox, the Intermedia programmers constructed a system that began with a conceptual application framework similar to the Apple Lisa [Appl83] or the XEROX Star [Xero82] environments to which they added full hypermedia capabilities. For a more detailed description of the underlying Intermedia architecture, refer to Meyrowitz, 1986.

2.3. The Applications

Five applications currently exist within the Intermedia framework: a text editor (*InterText*), a graphics editor (*InterDraw*), a scanned-image viewer (*InterPix*), a three-dimensional object viewer (*InterSpect*), and a timeline editor (*InterVal*). Any number of documents of different types may be open on the desktop at one time, along with the folders containing the documents. These applications conform as closely as possible to the Macintosh interface standards detailed in Apple, 1985. Both programmer-level tools and well-defined user interface concepts contribute to the high degree of consistency exhibited across all Intermedia applications.

The InterText word-processing application is similar to Apple's MacWrite [Appl84a], but with *style sheets* for formatting text rather than MacWrite-style rulers. Using style sheets, the user can define a set of styles for a particular document (for example, paragraph, title, subtitle, indented quote, numbered point) and apply those styles to any unit of text between two carriage returns, called an *entity*. When the user edits the definition of a style, all the entities to which that style is applied reformat accordingly.

With InterDraw, a structured graphics editor similar to Apple's MacDraw [Appl84b], users can create two-dimensional illustrations by selecting tools from a palette attached to each InterDraw window.

InterPix is a utility program that displays bitmap images entered into the system using a digitizing scanner. These images can be cropped, copied, and pasted into InterDraw documents. The InterPix application is being extended to provide full bitmap editing capabilities.

Like InterPix, InterSpect is a viewer rather than an editor. It converts files containing three-dimensional data points into three-dimensional representations of that data. Users can manipulate the three-dimensional image by rotating it, zooming in or out, or hiding parts of the model.

The fifth application, InterVal, provides interactive editing features for creating chronological timelines. As the user enters pairs of dates and labels, the application formats them on a vertical timeline according to user-defined styles. As with a charting package, the display of the data is determined by a certain set of parameters that the user can modify.

2.4. Hypermedia Functionality

The hypermedia functionality of the Intermedia system is integrated into each application so that the actions of creating and traversing links may be interspersed with the actions of creating and editing documents.

The act of making links between Intermedia documents has been modeled as closely as possible on the Smalltalk/Macintosh *copy/paste paradigm* [Appl85, Gold84] in an effort to fit the link-making process into a conceptual model already familiar to users. If links are to be made frequently, they must be a seamless part of the user interface. The smooth integration of this new function into an already ingrained user interface model is apparent in the sample session later in this section.

Unlike some other hypertext or hypermedia systems that allow links to be made only to entire documents [Know86 and Shne86], Intermedia allows users to create links from a specific location in one document to a specific location in another document. These "anchor points" in the documents are called *blocks*. In designing the Intermedia linking functionality, one of the development team's design goals was to allow anything that could be selected to be made into an anchor for a link. The size of a block, therefore, may range from an entire document to an insertion point, depending on the selection region a user identifies as the block's extent. For example, in an InterText document, a block might consist of an insertion point, a single character, a word, a sentence, two paragraphs, and so forth. Small marker icons are placed near the source and destination blocks to indicate the existence of a link.

To help manage a large corpus of linked documents, links and blocks are assigned a set of descriptive *properties*. Some of these, like user name and creation time, are assigned automatically, while other properties are user-defined. Users access and edit link and block property information through *property sheet dialogs*. A dialog contains a field in which the user enters a one-line *explainer*, similar to the subject field in some electronic-mail programs. Link explainers are particularly important from a reader's perspective. If a single block has more than one link emanating from it, users choose the path they wish to follow from a list of link explainers presented in a dialog box.

Property sheets also contain fields for adding keywords. Although still under development, these keywords, along with the default information assigned to links and blocks, will provide users with a mechanism for searching the document corpus. The result of a keyword search will be a list of explainers associated with all the blocks or links meeting the search criteria. Each item in the list will automatically be linked to its corresponding block, or, in the case of links, to the corresponding source block of each link. For example, a student would be able to search for all the links created by a particular professor after a certain date that contain the keywords "Browning" and "Dramatic Monologue."

Some users may want to enter both block and link properties during the link-creation process, while others may not want to take the time to fill them in at all. Property sheets are *not* automatically presented to the user by default. Intermedia provides a *Viewing Specifications* dialog as a utility for setting user preferences. To change the default behavior, users alter settings in the Viewing Specifications dialog, turning link and/or block creation to "verbose." With the verbose setting, the property dialogs are presented to the user immediately upon completion of block or link creation. If users have created links with the "fast" rather than the "verbose" setting, they can still edit block and link properties after a link has been established, by selecting a marker icon and choosing the "Link Properties" or "Block Properties" command from the menu.

Link and block properties help to manage complexity within the Intermedia environment, but the notion of *context* is even more crucial. In some systems, links are global; all links are available at all times to all users. In such systems, links become an integral part of the documents. In Intermedia, block and link information is not stored within individual documents, but rather is superimposed on them. *Webs* are provided to maintain the block and link information, allowing users to work within their own context undistracted by documents, blocks, and links created by others sharing the same computing resources. In the future, webs will also serve as the focus for filtering operations. Opening a web causes a particular set of blocks and links to be imposed on a set of documents for as long as the web is open. Thus, webs allow different users to impose their own links *on the same document set.* For example, an academic department might purchase all of Shakespeare's works in electronic format. Rather than duplicating every work each faculty member wants to link to, Intermedia's webs allow multiple users to create their own links to the same documents, without having to see each other's links. Although only one context may be viewed at a time, users can easily switch contexts by closing one web and opening another. Of course, if users *do* want to work together, any number may share a single web.

In addition to webs, Intermedia also has a system of user access rights that helps to manage multiple users sharing large bodies of connected material. Due to the hypermedia functionality of Intermedia, the access-right scheme is slightly more complex than that of other UNIX environments where users may have either *read permission* or *write permission* to files and directories. Intermedia adds *annotation permission* to the other two forms of access rights. This allows students to add links to a document that they are not allowed to edit.

2.5. Common User-Interface Concepts

Several user-interface concepts stressed throughout Intermedia enable users to learn new applications quickly and predict the behavior of features they have never used before. Like the copy and paste operations in the Macintosh and Smalltalk programs, some operations in the Intermedia system behave identically across all applications. The linking functionality is a prime example. Users can specify a selection region in any document, choose the "Start Link" command from the menu, define another selection region in any other document, regardless of type, and choose one of several "complete link" commands. Likewise, to follow a link, a user exploring a web can select a marker icon in any type of document, and choose the "Follow" command from a menu. As a shortcut, a user can double-click on a marker icon to initiate the Follow, just as he or she might double-click on an icon in a folder to open a document. Since following a link usually entails opening a document, we anticipated that users would *expect* to be able to follow a link by double-clicking on the marker icon. In a system that encourages rapid transitions between applications, it is essential to limit the amount a browser must learn in order to be a successful user of the system, in part by capitalizing on those conventions he or she may already be familiar with.

Other features in the system, while not exactly identical to one another, are conceptually similar. Most applications, for instance, allow users to control the format or the display characteristics of data. The interface techniques for conceptually similar operations have been designed to capitalize on the similarities. One such example is the *style paradigm* [Inte86, Smit82]. Styles are sets of properties or characteristics that govern the appearance of data within a document. Users can define or modify a style by filling out or editing a "form" called a *style sheet* (sometimes referred to as a *property sheet*). Both the InterText application and the InterVal application contain style sheets that can be used to specify different text formats such as paragraphs, indented quotes, lists, and titles, or different timeline formats such as position of dates relative to tick marks and position of labels relative to dates. In the graphics editor, different styles may be applied to shapes such as line width, pen style, or fill style. Although not yet implemented, InterDraw will also allow users to "borrow" the style of one shape and "apply" it to any others. This means that the combination of line width, pen style, and fill style can be defined once and then applied as a unit to any existing objects or to any new objects. The concept of styles will also be included in the InterSpect three-dimensional viewer in the future. In this case, users will be able to define a set of viewing parameters that specify a particular camera position. Just as styles in the text editor do not alter the content, but simply the presentation, of the text, styles in InterSpect will be used to define the manner in which the data is viewed. By storing all presentation parameters in style sheets, it is possible to substitute styles with the same name but different parameters for all types of data in the system.

Related to the use of styles is the frequent use of *palettes* —sets of controls attached to a document window. Along with style-sheet dialogs, palettes provide a means for defining and applying styles. In InterText, for example, all the styles defined for a particular document are stored in a style palette. Two mouse clicks are all that is necessary either to change the style of an existing text entity or to change from one style to another before beginning a new entity. With a large screen and the ability to have a large number of windows open at one time, it is essential that all the tools needed for common operations be close at hand rather than in the menus. When they are not needed, however, all palettes can be hidden from view to unclutter the screen and to improve the manner in which material is presented to a person browsing through the system.

Another example of a standard user interface concept that permeates the system and is made possible in a workstation environment with virtual memory capabilities is the use of "infinite" *undo* and *redo* commands. Instead of being able to retract only the last action performed, the user is able to undo the effects of all actions performed since the last time a document was saved. Any single action or set of actions that the user has undone may then be redone. There are no content-modification actions in any Intermedia applications that cannot be undone. This fosters a sense of security in users, permitting them to experiment freely with their document with the knowledge that they can return it to its former state at any time.

To aid programmers in incorporating the standard features mentioned above (such as "undo" and "redo") consistently throughout all Intermedia applications, the software-development team first extended Apple's object-oriented application framework, MacApp, and then implemented two general-purpose, programmer-level *building blocks* within the extended framework. The building blocks, one for text and one for graphics, allow programmers to reuse text and graphics objects either "as is" or in slightly modified forms, cutting down on duplication of programming effort as well as enforcing consistent user-interface techniques. With only MacApp and the building blocks, programmers would be able to implement applications that conform to the Macintosh user-interface standards; however, the requirements for the Intermedia system include a uniform method of linking together the contents of documents created with any application within the system. To meet this requirement, the development team extended MacApp to include a set of features necessary for supporting linking functionality. In this way, applications not only inherit window, menu, mouse, and other generic application functionality from MacApp, but also inherit linking functionality from the MacApp extensions, and text and graphic editing functionality from the building blocks.

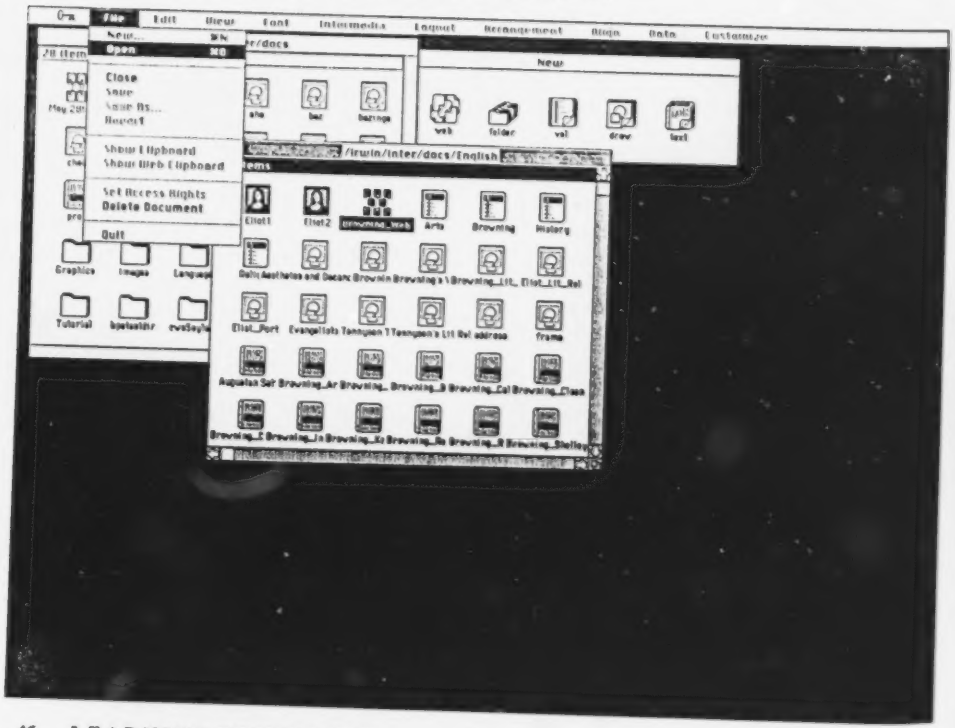
2.6. Sample Session

To illustrate the user-interface features and the linking functionality discussed above, this section of the paper takes the reader on a tour of the system. The tour is designed to simulate the interaction that takes place during a hands-on Intermedia session. The figures should aid in visualizing the system, while the text supplies the action.

2.6.1. Figure 1

As you can see from Figure 1, the Intermedia desktop includes a window manager, a graphical folder system, a menu bar, and a mouse interface. The contents of the folders reflect the underlying hierarchical UNIX directory structure. Unlike the Macintosh, Intermedia does not store application icons in the same folders with documents; instead, they are stored with several other special-purpose tools in an application, or *New*, window that you can see in the upper right corner of the screen. The reason for this is twofold. First, users do not have to search through folders to find the applications. Even if the New window is hidden from view by overlapping windows, selecting the **New** command from the **File** menu reveals it. Second, in a networked environment, it is easier to have a single set of applications in an agreed-upon place that can be maintained and updated by a system administrator.

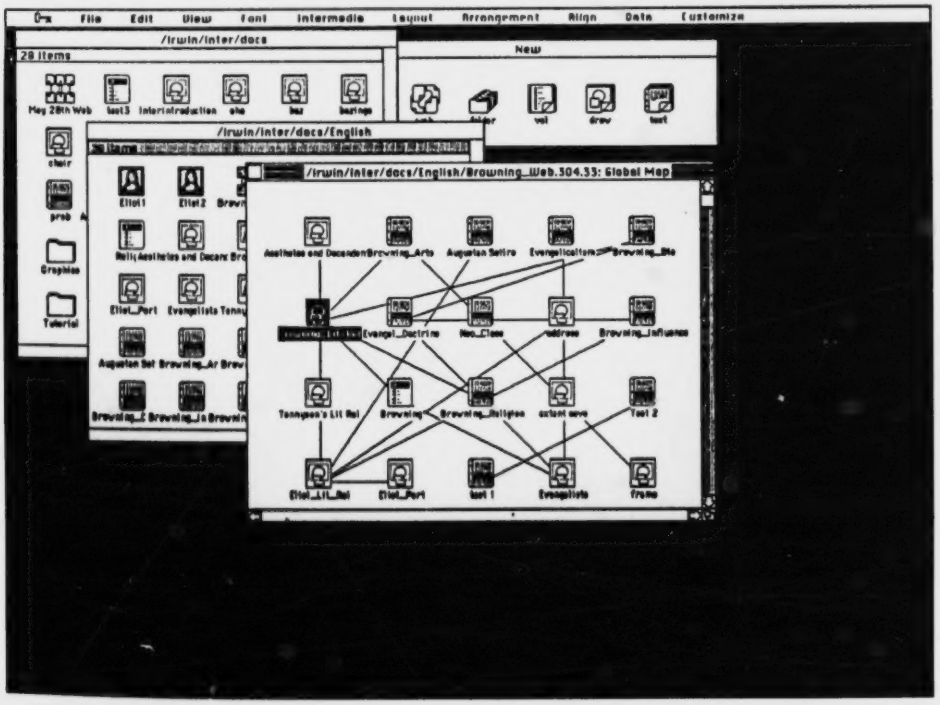
The contents of the active folder in Figure 1 represent a number of documents of different types (two InterPix, four InterVal, ten InterDraw, and seventeen InterText documents) that are samples from a larger corpus of material called *Context 32: A Web of English Literature* [Land86], created for the course "English 32: A Survey of English Literature from 1700–Present." Any of these document icons can be selected, opened, and edited. In order to follow or create links, however, we must first define a context by opening an existing web or by creating a new one. If a web is not open, we can still open and edit the documents; however, no link or block information will be visible. Rather than beginning a new web, we select the icon titled **Browning Web** and choose the **Open** command from the **File** menu.



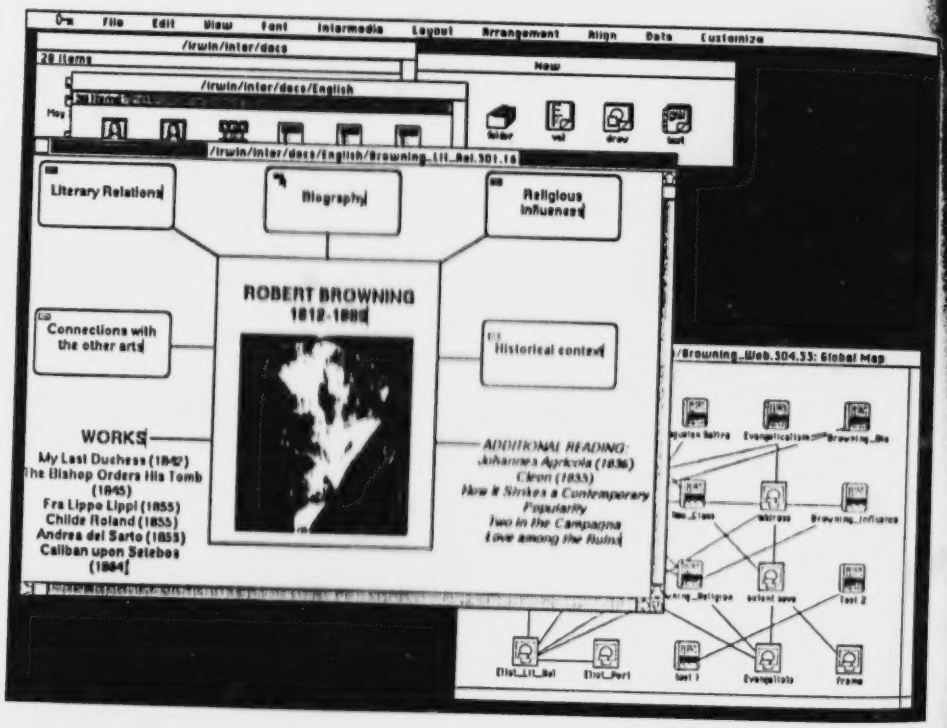
2.6.2. Figure 2

A *global map,* or visual representation of the link structure, opens on the screen. Global maps are analogous to road maps of an entire country. They depict every document in a corpus; one line between two document icons indicates that at least one link exists between the documents. In a large corpus, users can see only a small portion of the map at any one time. The global map provides a broad overview; just as a traveler can see at a glance which cities are the biggest hubs by the number of roads emanating from them, an Intermedia user can see which documents are central to the corpus by observing the number of links connected to them.

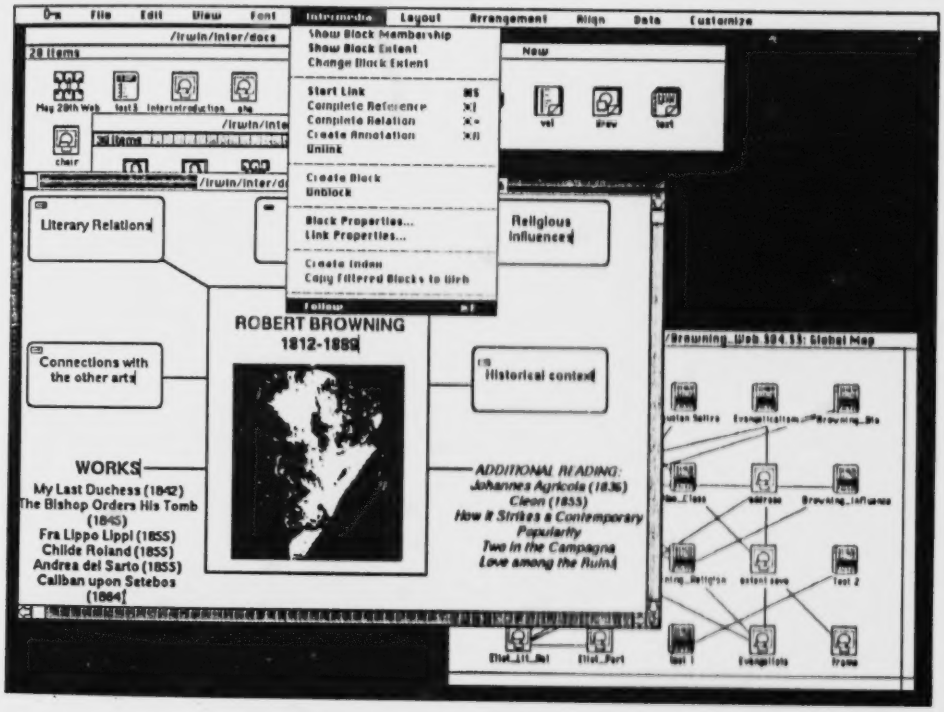
An examination of the global map open in Figure 2 reveals that the InterDraw document **Browning's Literary Relations** has the greatest number of connections and would therefore be the most appropriate starting point for our exploration. Once a user selects an entry point into the corpus, the equivalent of a specialized street map is necessary. *Local maps*, not shown in this scenario, serve this purpose by *tracking* the user's progress through the corpus and displaying only the currently active document and the links that emanate from it. When the user activates a different document, either by following a link or by opening one from a folder, the local map updates to display the new current document and its direct predecessor and successor links.



2.6.3. Figure 3
After double-clicking on the "Browning's Literary Relations" icon to open it, an InterDraw document containing a diagram with an image of Robert Browning opens. Notice that the three boxes in the top row of the diagram contain marker icons (arrows enclosed in boxes). These markers indicate the existence of links. To follow one of the links shown, we first select the marker above the text **Biography** by clicking once on it.

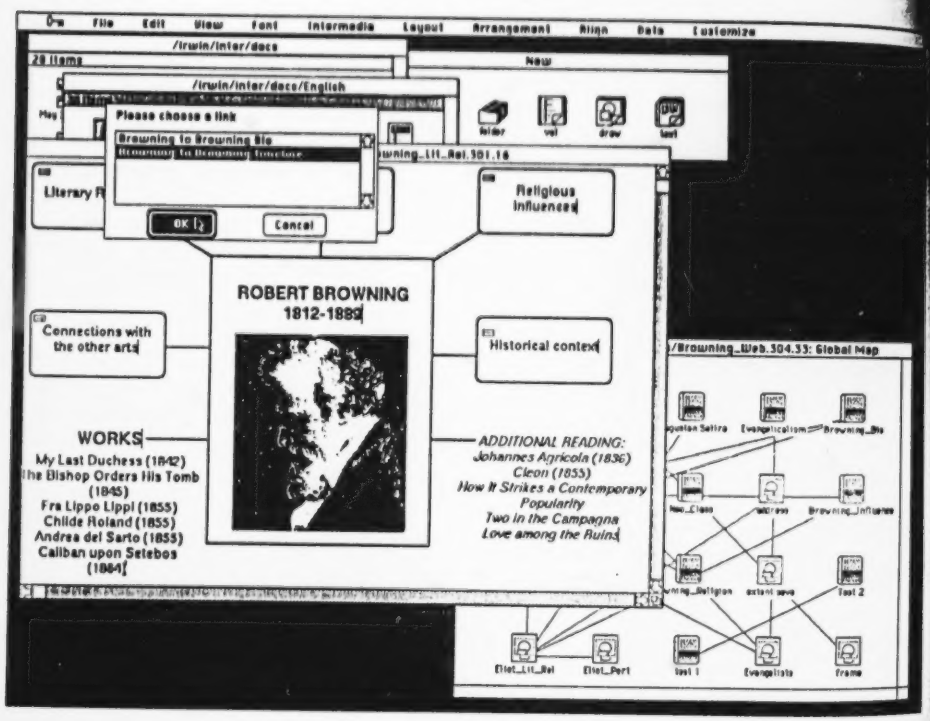


2.6.4. Figure 4
After selecting a marker icon, we pull down the **Intermedia** menu and select the **Follow** command.



2.6.5. Figure 5

We immediately learn, from the appearance of a dialog box, that the marker we selected has two links emanating from it: one leading to a biography of Browning and the other leading to a timeline. We select the **Timeline** explainer in the dialog box and click on the **OK** button to confirm our selection.

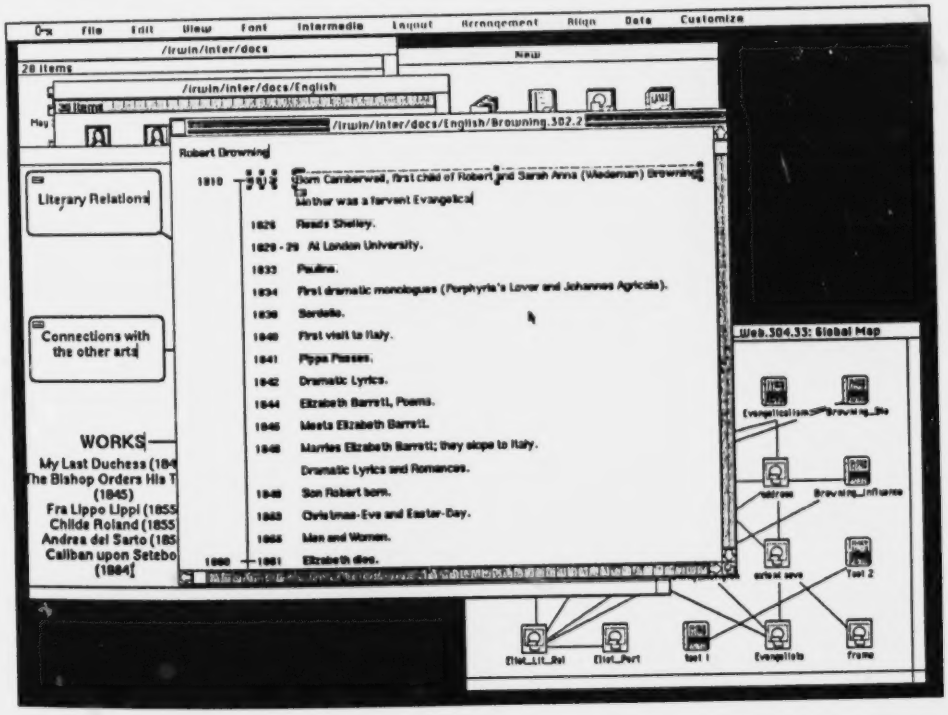


2.6.6. Figure 6

The traversal of the link is now complete and the timeline document, which lists important events in Browning's life, appears. The author of Browning's timeline chose to leave the events clustered in 50-year intervals so that a majority of the events could be viewed in the window at once. In other words, all the events between 1810 and 1859 are clustered under the tickmark by 1810, and all the events between 1860 and 1909 are clustered under the tickmark by 1860.

Notice that when a link is traversed, Intermedia automatically highlights the *extent* of the block at the other end of the link, indicating a particular scope of information to the reader. In this case our attention is drawn to the first event in the timeline.

We can now opt to recluster the timeline, but instead, we notice another marker icon above the second event on the timeline and decide to follow the link. We do so by double-clicking on the **Marker**.

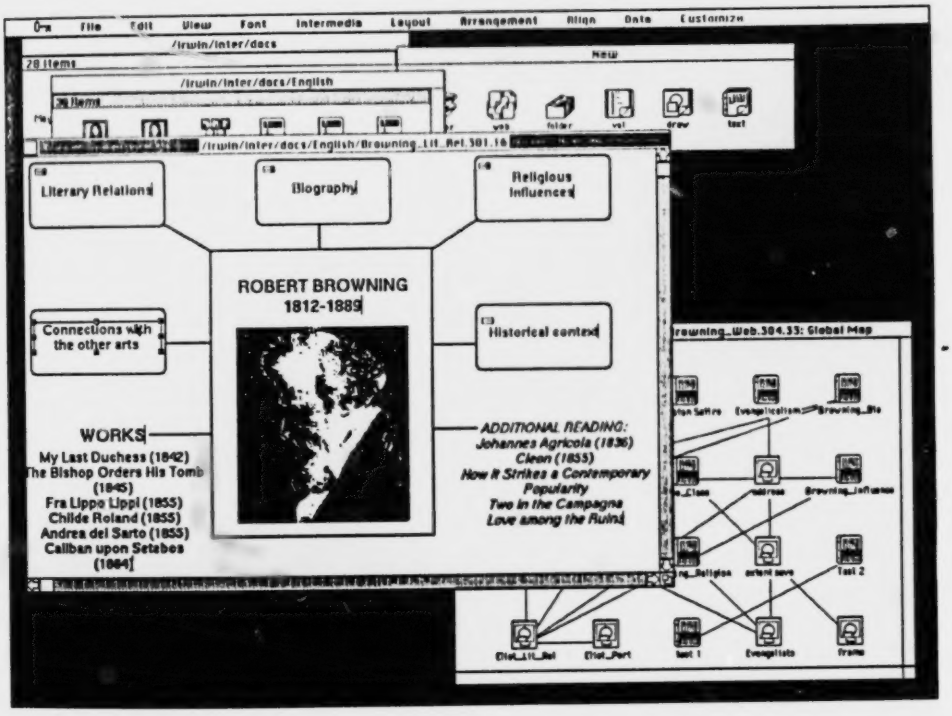


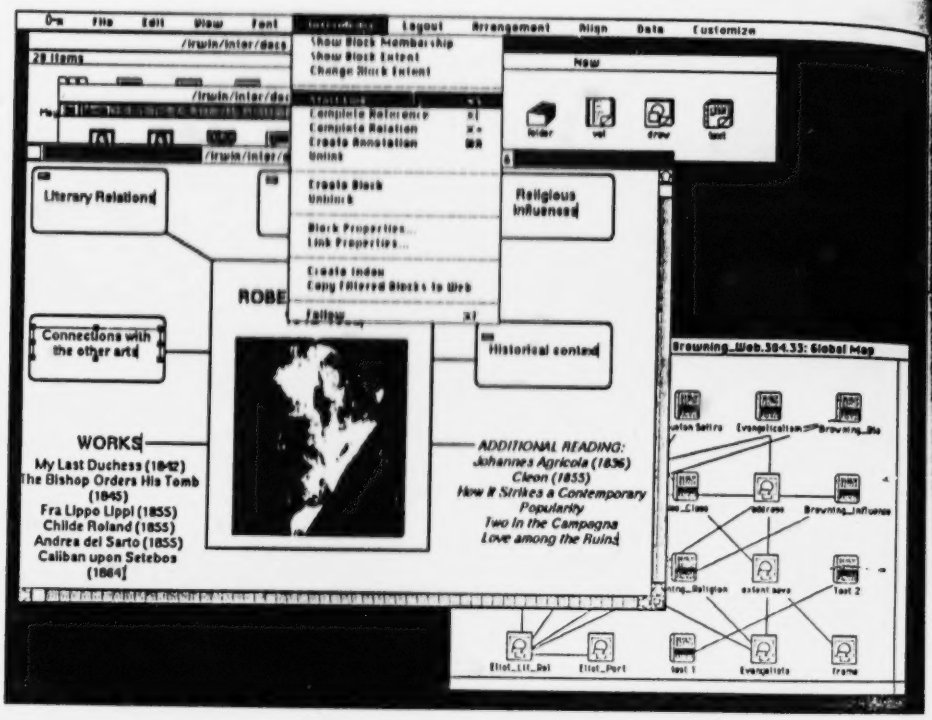
2.6.7. Figure 7
Another InterDraw document opens. Notice that no dialog box appears this time. When we select a marker icon that has only one link emanating from it, the document at the other end of the link opens immediately, since there are no alternate paths to follow.

Like the portrait of Browning in Figure 2, the illustration in Figure 7 was entered into the system using a scanner. The bitmap was then displayed by the InterPix application, and was cropped and pasted into this InterDraw document. After looking at the picture, we decide to close it by clicking in the *close box* located in the window's upper left corner. Next we close the InterVal timeline.

2.6.8. Figures 8 and 9

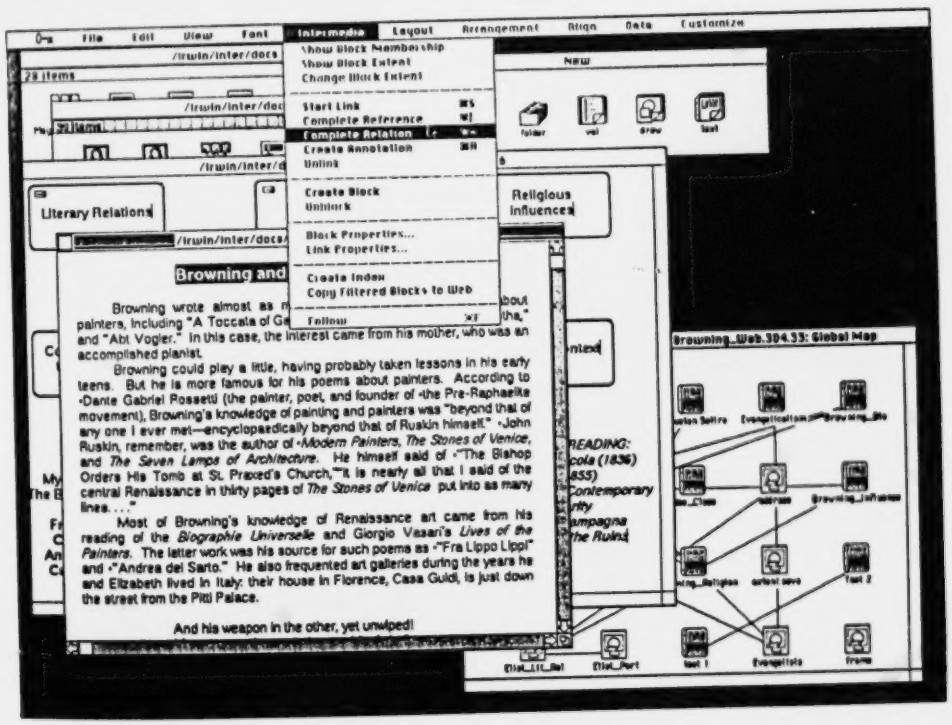
Now we are back to the original Browning diagram. After our brief exploration, we decide to fill in the diagram by creating a new link. The first step in creating a link involves defining a block to serve as the anchor for the link. We select the text "Connection to Other Arts" as the source block of the link and choose "Start Link" from the **Intermedia** menu. While a link is pending, we may perform any number of other actions unrelated to link-making. As with the "Copy" operation common to all Macintosh-like applications, the **Start Link** operation is completely modeless.




File
Edit
View
Font
Layout
Arrangement
Align
Data
Customize
Show Block Membership
Show Block Extent
Change Block Extent
Create Block
Unblock
Block Properties...
Link Properties...
Create Index
Copy Filtered Blocks to Web
New
Literary Relations
Religious Influenced
Connections with the other arts
Historical context
WORKS
My Last Duchess (1842)
The Bishop Orders His Tomb (1845)
Fra Lippo Lippi (1855)
Childe Roland (1855)
Andrea del Sarto (1855)
Caliban upon Setebos (1864)
ADDITIONAL READING:
Johannes Agricola (1836)
Cleon (1855)
How It Strikes a Contemporary
Popularity
Two in the Campagna
Love among the Ruins
Browning_Web.384.33: Global Map

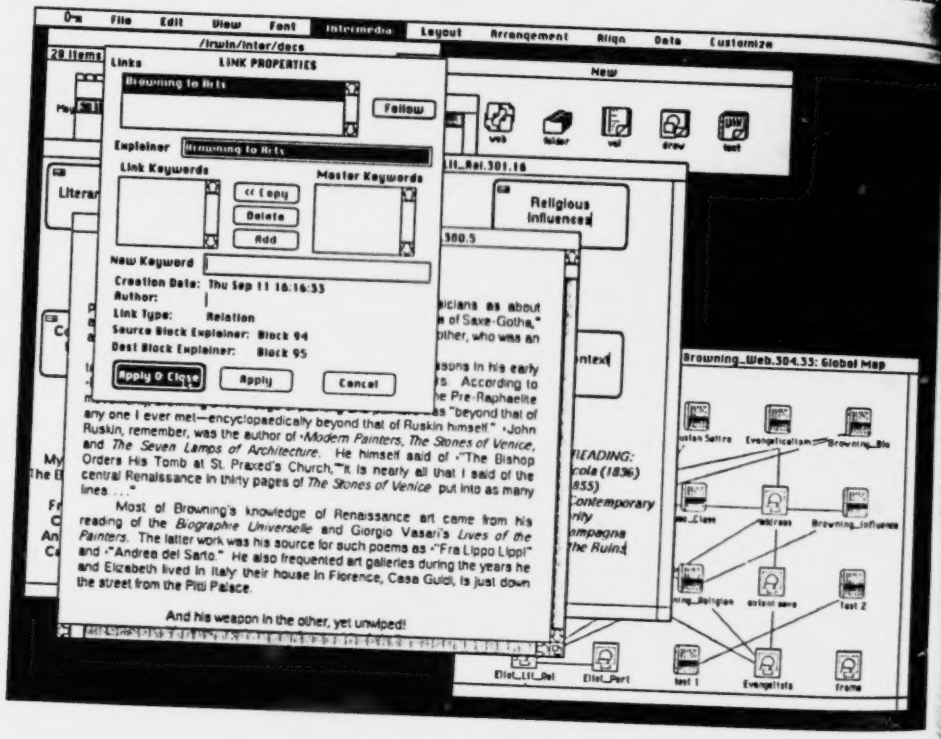
2.6.9. Figure 10

Before completing the pending link, we return to the folder (not pictured) and open an InterText document called **Browning Arts**. Once the text is displayed, we select the heading of the document as the destination block of the link, as shown in Figure 10. To complete the link we choose **Complete Relation** from the **Intermedia** menu. You will notice two different "complete" commands in the menu. These are similar to each other in function, but each creates a different type of link. The "Complete Relation" command, which we chose, indicates a primary path, whereas the "Complete Reference" command signifies that the information contained in the source block is of more importance than the information contained in the destination block.


File
Edit
View
Font
Intermedia
Layout
Arrangement
Align
Data
Customize
Show Block Membership
Show Block Extent
Change Block Extent
Start Link
Complete Reference
Complete Relation
Create Annotation
Unlink
Create Block
Unblock
Block Properties...
Link Properties...
Create Index
Copy Filtered Blocks to Web
Follow
Literary Relations
Religious Influences
Browning and
Browning wrote almost as
painters, including "A Toccata of
and "Abt Vogler." In this case, the interest came from his mother, who was an accomplished pianist.
Browning could play a little, having probably taken lessons in his early teens. But he is more famous for his poems about painters. According to •Dante Gabriel Rossetti (the painter, poet, and founder of •the Pre-Raphaelite movement), Browning's knowledge of painting and painters was "beyond that of any one I ever met—encyclopaedically beyond that of Ruskin himself." •John Ruskin, remember, was the author of •Modern Painters, The Stones of Venice, and The Seven Lamps of Architecture. He himself said of •"The Bishop Orders His Tomb at St. Praxed's Church,""it is nearly all that I said of the central Renaissance in thirty pages of The Stones of Venice put into as many lines. . . ."
Most of Browning's knowledge of Renaissance art came from his reading of the Biographie Universelle and Giorgio Vasari's Lives of the Painters. The latter work was his source for such poems as •"Fra Lippo Lippi" and •"Andrea del Sarto." He also frequented art galleries during the years he and Elizabeth lived in Italy: their house in Florence, Casa Guidi, is just down the street from the Pitti Palace.
And his weapon in the other, yet unwiped!
Browning_Web.304.33: Global Map

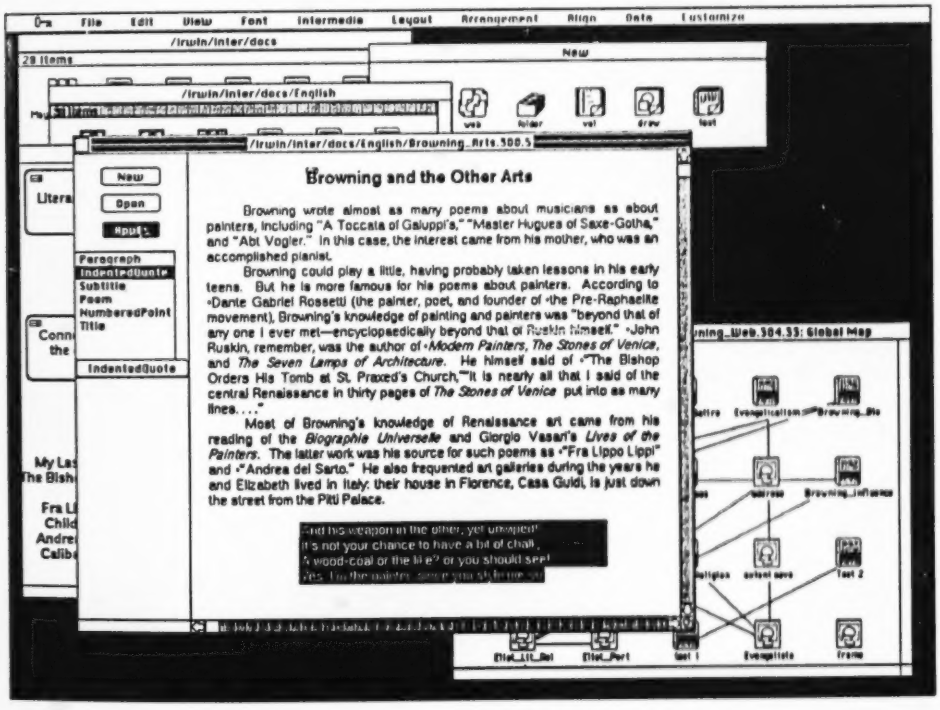
2.6.10. Figure 11

After we issue the "complete" command, a link property dialog appears, allowing us to fill in descriptive information about the link. We replace the default text, "Link 35," with the more meaningful explainer, **Browing to Arts** shown in Figure 11. Before this particular session, we changed the default setting for link creation in the Viewing Specifications dialog so that a link property dialog would automatically be opened each time we create a link.

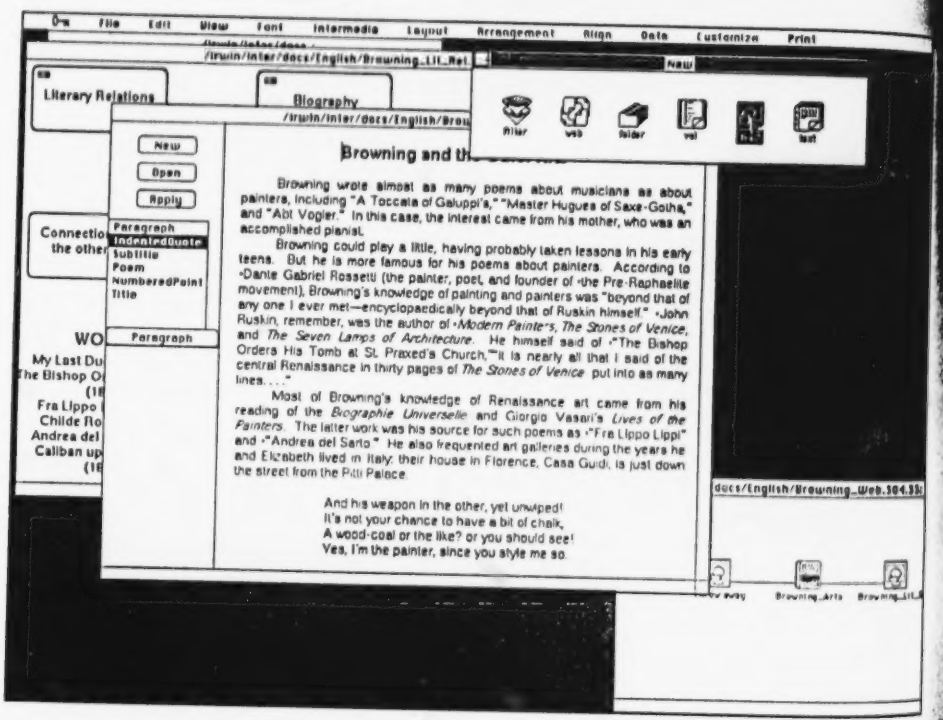

File
Edit
View
Font
Intermedia
Layout
Arrangement
Align
Data
Customize
/irwin/inter/docs
LINK PROPERTIES
Browning to Arts
Follow
Explainer
Link Keywords
Master Keywords
<< Copy
Delete
Add
New Keyword
Creation Date: Thu Sep 11 16:16:33
Author:
Link Type: Relation
Source Block Explainer: Block 94
Dest Block Explainer: Block 95
Apply & Close
Apply
Cancel
New
Religious Influences
Browning_Web.304.33: Global Map
Most of Browning's knowledge of Renaissance art came from his reading of the Biographie Universelle and Giorgio Vasari's Lives of the Painters. The latter work was his source for such poems as "Fra Lippo Lippi" and "Andrea del Sarto." He also frequented art galleries during the years he and Elizebeth lived in Italy: their house in Florence, Casa Guidi, is just down the street from the Pitti Palace.
And his weapon in the other, yet unwiped!

2.6.11. Figure 12

Once a link is established, both ends are indicated with markers. In some applications, these markers are enclosed in boxes for better visibility, but the markers themselves are identical in all applications. Before making another link, we add a quotation to the "Browning Arts" document. After typing in the text of the quotation, we select the text, display the *style palette* with a menu command, choose the **Indented Quote** style from the palette and click on the **Apply** button, also located in the palette. With a simple text selection followed by a palette selection and a button press, we have easily reformatted the text without resorting to the more complex series of steps necessary in other word-processing programs.


File
Edit
View
Font
Intermedia
Layout
Arrangement
Align
Data
Customize
/irwin/inter/docs
29 Items
New
/irwin/inter/docs/English
/irwin/inter/docs/English/Browning_Arts.500.5
New
Open
Paragraph
IndentedQuote
Subtitle
Poem
NumberedPoint
Title
IndentedQuote
Browning and the Other Arts
Browning wrote almost as many poems about musicians as about painters, including "A Toccata of Galuppi's," "Master Hugues of Saxe-Gotha," and "Abt Vogler." In this case, the interest came from his mother, who was an accomplished pianist.
Browning could play a little, having probably taken lessons in his early teens. But he is more famous for his poems about painters. According to Dante Gabriel Rossetti (the painter, poet, and founder of the Pre-Raphaelite movement), Browning's knowledge of painting and painters was "beyond that of any one I ever met—encyclopaedically beyond that of Ruskin himself." John Ruskin, remember, was the author of *Modern Painters*, *The Stones of Venice*, and *The Seven Lamps of Architecture*. He himself said of "The Bishop Orders His Tomb at St. Praxed's Church," "It is nearly all that I said of the central Renaissance in thirty pages of *The Stones of Venice* put into as many lines. . . ."
Most of Browning's knowledge of Renaissance art came from his reading of the *Biographie Universelle* and Giorgio Vasari's *Lives of the Painters*. The latter work was his source for such poems as "Fra Lippo Lippi" and "Andrea del Sarto." He also frequented art galleries during the years he and Elizabeth lived in Italy; their house in Florence, Casa Guidi, is just down the street from the Pitti Palace.
Web.584.33: Global Map

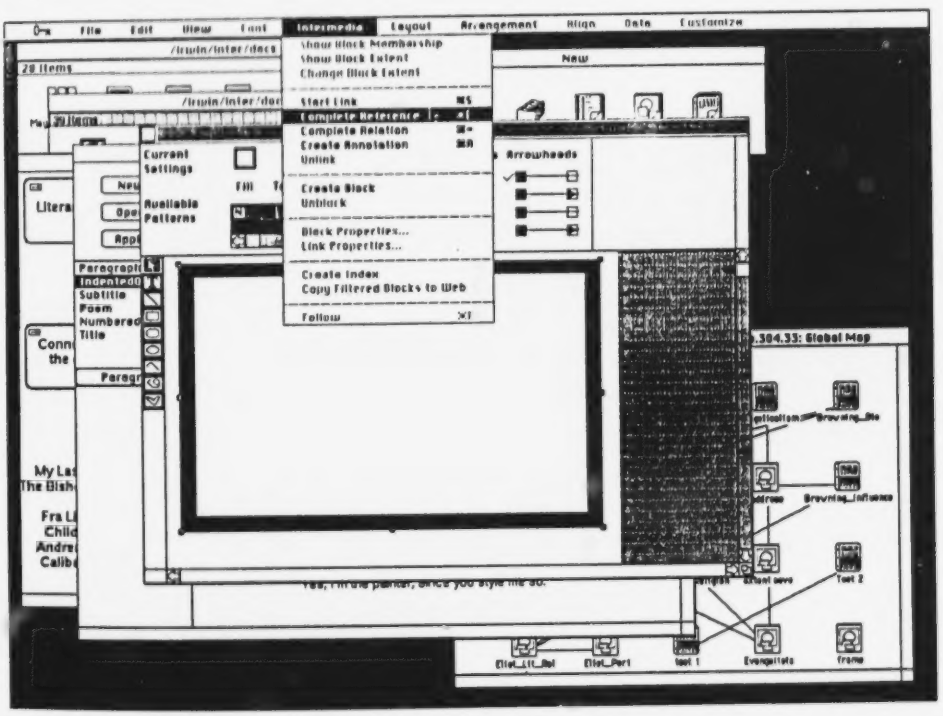
2.6.12. Figure 13
Now we decide to make another link to a new InterDraw document that may contain a painting in the same stylistic genre as Browning's poetry. The destination block of the previous link is appropriate for the source block of our new link, so we select the existing marker by clicking on it and then we choose **Start Link** from the menu (not pictured). Now, to create a new document, we activate the **New** window and double-click on the **InterDraw** icon.


File
Edit
View
Font
Intermedia
Layout
Arrangement
Align
Date
Customize
Print
Literary Relations
Biography
New
Open
Apply
Paragraph
IndentedQuote
Subtitle
Poem
NumberedPoint
Title
Paragraph
Browning and th
Browning wrote almost as many poems about musicians as about painters, including "A Toccata of Galuppi's," "Master Hugues of Saxe-Gotha," and "Abt Vogler." In this case, the interest came from his mother, who was an accomplished pianist.
Browning could play a little, having probably taken lessons in his early teens. But he is more famous for his poems about painters. According to Dante Gabriel Rossetti (the painter, poet, and founder of the Pre-Raphaelite movement), Browning's knowledge of painting and painters was "beyond that of any one I ever met—encyclopaedically beyond that of Ruskin himself." John Ruskin, remember, was the author of Modern Painters, The Stones of Venice, and The Seven Lamps of Architecture. He himself said of "The Bishop Orders His Tomb at St. Praxed's Church," "It is nearly all that I said of the central Renaissance in thirty pages of The Stones of Venice put into as many lines. . . ."
Most of Browning's knowledge of Renaissance art came from his reading of the Biographie Universelle and Giorgio Vasari's Lives of the Painters. The latter work was his source for such poems as "Fra Lippo Lippi" and "Andrea del Sarto." He also frequented art galleries during the years he and Elizabeth lived in Italy; their house in Florence, Casa Guidi, is just down the street from the Pitti Palace.
And his weapon in the other, yet unwiped!
It's not your chance to have a bit of chalk,
A wood-coal or the like? or you should see!
Yes, I'm the painter, since you style me so.
docs/English/Browning_Web.304.35

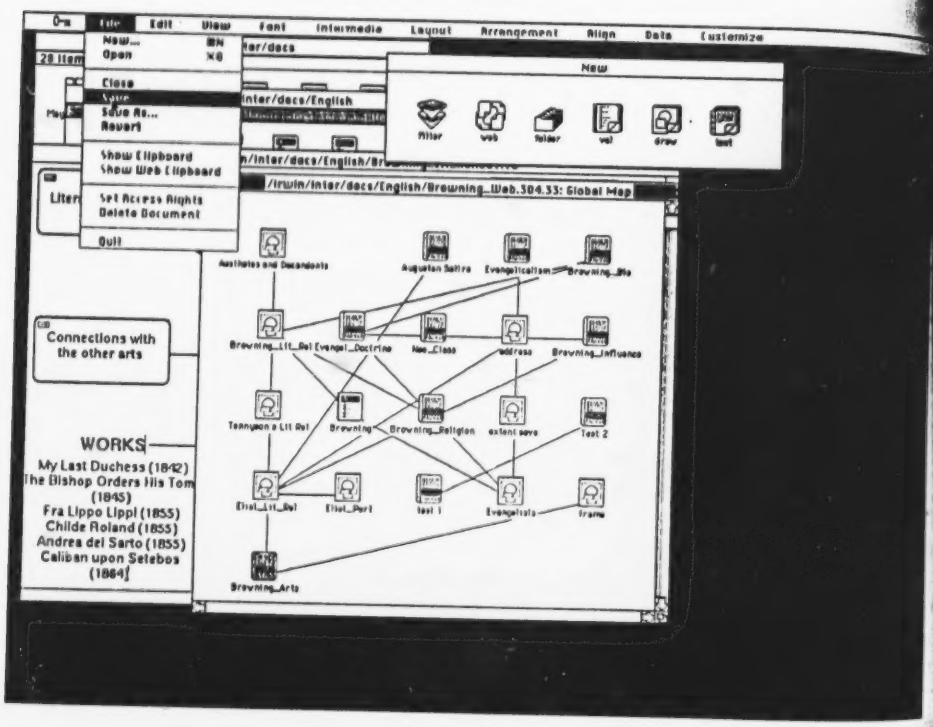
2.6.13. Figure 14

After the new document opens, we see that there is no appropriate painting, so we draw a picture frame in which we will paste a scanned image at some later time. For the drawing, we use tools from both the tool and style palettes attached to the window. After the drawing is complete, we may hide one or both of the palettes. When a document is saved, the size of the window and the state of the palettes is preserved along with all editing changes.

Remember that a link is still pending. In the absence of an appropriate painting, we select the empty frame as the destination block of the pending link and choose **Complete Reference** from the **Intermedia** menu. This time, a new marker will be added only to the destination block, since the source block is already marked.



2.6.14. Figure 15
Before we end our session, we activate the global map window and choose **Save** from the **File** menu. This begins a sequence that prompts the user to determine whether to save the new links stored in the web, the documents that contain new blocks or links, new documents, and documents that have been edited.



The tour through Intermedia illustrates the functionality of but one of several hypermedia systems under development today. We have designed a user interface that we believe creates an easy-to-use but powerful environment for both the reader and the author. That user interface was the result of much enumeration of possibilities, heated discussion, comparison of choices, and consideration of what could be accomplished given today's technology.

The second half of this paper discusses the issues that we and other designers faced in determining the hypermedia capabilities of our systems. To begin the discussion, we first examine what the most elementary read-only hypermedia system might look like and what design issues are involved. We then move to a system in which a single user may edit the multimedia corpus. Finally, we explore the issues of multiple users accessing a shared set of documents.

A System of Read-Only Documents

The simplest hypermedia system is one in which a single user at a stand-alone workstation examines read-only documents and creates and follows links among these documents. This section explores some of the concerns that appear in designing such an environment, including the issues of providing appropriate (1) contexts for webs of links,(2) anchors for individual links, and (3) visual representations of the linked corpuses.

3.1. Providing Contexts

When an author links to a destination document that contains links from previous sessions, either all of the links of the destination document can be incorporated into the set of links on which the author is working, or the system can display the destination document as if it has no links at all.

In the first situation, the author would automatically inherit all of the connections anchored in the destination document; all of the links would exist in the same "plane" [Figure 16]. By making links, an author would thereby provide bridges to other worlds of links. This bridge is appealing in systems where the fundamental goal is to encourage the maximum interconnectedness of information. This would, however, prevent authors from presenting a limited corpus.

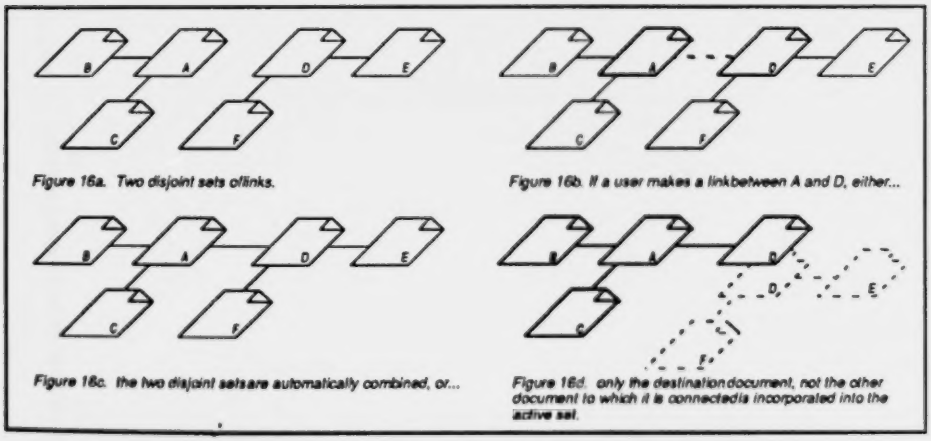


Figure 16a. Two disjoint sets oflinks.

Figure 16b. If a user makes a linkbetween A and D, either...

Figure 16c. the two disjoint setsare automatically combined, or...

Figure 16d. only the destination document, not the other document to which it is connectedis incorporated into the active set.

The second option solves this problem by enabling the user to partition links into private worlds, or separate contexts. Such contexts would be useful to a scholar who wants to organize material for a variety of research projects, each of which may reference the same document.

Intermedia designers opted for the second solution with a construct called *webs*. Until the user opens a web, all opened documents appear without any links. To view documents with the links that belong to a particular web, a user opens a web and then opens one or more of its documents. Although other webs may also reference the documents, only the links that were made in the current web are available. As a result the user does not have to sift through the connections of multiple contexts.

3.2 Linking

3.2.1 The Link Anchors

Textual anchors are currently limited to insertion points and are represented internally by only one pointer. In Intermedia graphics applications, the user specifies an anchor by selecting any number of graphics objects, regardless of their proximity, and issuing the appropriate command. A graphics anchor is represented internally by a list of objects included in the anchor rather than by the start and end points of a sequence.

Anything from entire documents to precise insertion points could serve as anchors for links, depending on the purpose of the system under development. Between these extremes lie other possibilities: a link might be anchored to any number of user-defined text segments or graphics objects or to a single word or graphics object.

The designers may make the decision between these alternatives by balancing implementation and functional concerns. A system that allows a user to link only to entire documents is easier to implement than the other options, but it does not allow the user to reference specific contents. Being able to link to insertion points is an improvement over linking to entire documents, and it involves a manageable increase in implementation complexity. With insertion points, however, references may be ambiguous. Some authors might place the link at the beginning of a reference to signal that what follows is important, while others might place the link at the end, as they are used to doing with conventional footnotes. If text segments or graphics objects could be the link anchor, the author would be able to indicate the reference clearly.

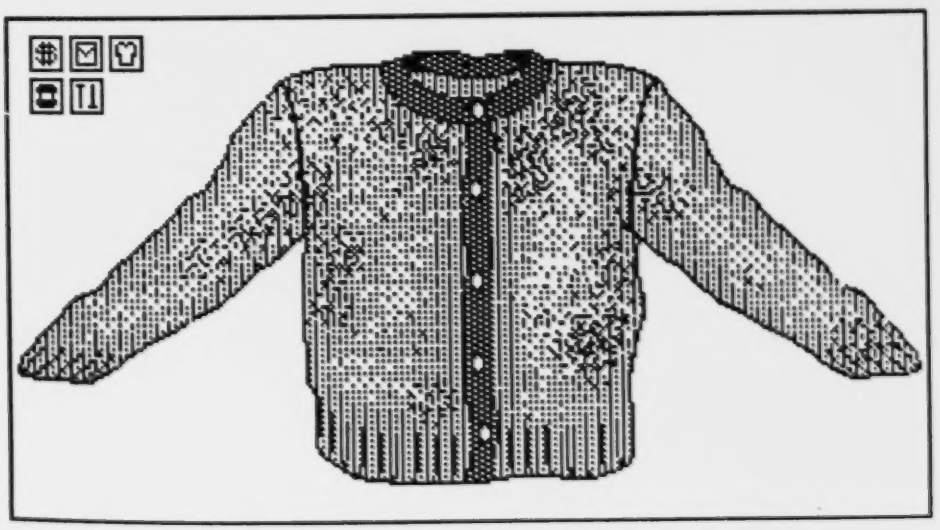
For some purposes, such as a system for indexing or for organizing encyclopedia information, yet another design might be appropriate. Such a system might provide automatic linking so that each link would be characterized by a single keyword on one end and an entire document on the other.

As designed, an Intermedia anchor may typically be any legitimate selection in that application, such as a user-defined sequence of characters or a combination of graphics objects. With this design, a textual anchor could be represented internally as pointers to the start point and end point in the underlying model. However, due to complications caused by providing text editing functionality (see Section 4), and time constraints in implementation, we were not able to implement this design fully in the first version of Intermedia.

3.2.2. Indicating Links

For users to be able to follow links, the system needs to indicate to the users where the links are anchored. The system could indicate the links in a number of ways, including marking the beginning, end, or extent of the anchor. The marking might consist of (2) a single icon at each anchor, where there may be more than one link per anchor; or (3) highlighting the extent of the anchor with font or color changes, outlining, or reverse video. The indication of links could be done automatically by the system or upon the user's request. In deciding how to indicate links, designers must consider the purpose of the system, the desirability of user control, and the potential problems of display clutter and ambiguity.

Icons are an appropriate way to mark links in some systems. Systems that require or allow anchoring links to insertion points would benefit from icons because they provide a tangible link anchor. Furthermore, icons can be designed to convey additional meaning such as whether they are the origin or destination of the link and whether the link leads to text or graphics. For example, imagine Penny Straker creating a hypermedia mail-order catalog of traditional hand-knit sweaters. She wants to link from a picture of a sweater to information on price, ordering, size, yarn, and the history of the sweater. If all of the icons had the same design, the shopper might have to follow each link to find the desired information. If, however, the icons were designed to indicate graphically the various kinds of information, the shopper would know which to follow [Figure 17].



However, there may be problems with excessive or ambiguous icons. Readers might find the icons intrusive, especially in a document with many links. With one icon per link, many icons might need to be displayed at the same location. For example, an article on Pennsylvania might mention Gettysburg. An author might have linked from Gettysburg to articles on Presidents Lincoln and Eisenhower, a description of the Battle of Gettysburg, and a copy of the Gettysburg Address. Four similar icons would be clustered at the same place, and the user would have no way to tell which one to follow. Even with the different icon design strategy mentioned above, the display could still become cluttered. Another strategy is to allow the user to control whether icons are displayed. The user could turn off the icon display to concentrate on the content of the document and subsequently turn it on to examine linked references. A third possibility is to limit each anchor to one icon, thereby uncluttering the display. However, because information might be lost, the system would need to provide more information about the links that emanate from an anchor. Such additional information might include how many links exist and to what kind of document each one leads.

Icons are sufficient for marking the ends of links when anchors are just insertion points. However, if anchors can themselves have contents, icons alone cannot give complete visual information about the anchor. A supplement to icons is the display of the extent of the anchor. Highlighting could consist of font or color changes, outlining boxes, or reverse video. Not only would such highlighting accurately show the extent of nonoverlapping anchors, but it could also show the individual extents of overlapping anchors [Figure 18].

IRIS is charged with two specific responsibilites. The first is to conduct and encourage experimentation in the development and innovative use of technology in education and scholarship. The second responsibility is to administer and manage the "Scholars' Workstation Project," a campus-wide effort directed toward

IRIS is charged with two specific responsibilites. The first is to conduct and encourage experimentation in the development and innovative use of technology in education and scholarship. The second responsibility is to administer and manage the "Scholars' Workstation Project," a campus-wide effort directed toward

IRIS is charged with two specific responsibilites. The first is to conduct and encourage experimentation in the development and innovative use of technology in education and scholarship. The second responsibility is to administer and manage the "Scholars' Workstation Project," a campus-wide effort directed toward

Although highlighting might provide some improvement over icons, it might still clutter the display. As with icons, the user could control when the anchor highlighting is displayed by toggling the appropriate switch. In addition, the system could provide a context-sensitive cursor that would invoke highlighting only when it moved to a point within an anchor.

Intermedia uses a combination of icons and anchor highlighting to show links and anchors. Each anchor has one icon, regardless of the number of links attached to it. In graphics documents, icons mark the existence of anchors, and highlighting indicates anchor extent. The user may select any number of icons and ask that their extents be highlighted. Currently, text-document anchors are limited to insertion points that are marked by icons. Because each icon may indicate more than one link, when the user selects an icon and gives the "Follow" command for an anchor with more than one link, Intermedia presents a list of the links that are attached to the anchor. The user can then choose which link to follow.

Plans for future versions of Intermedia include allowing the user to control icon display. Text will mirror graphics by allowing the user to attach links to arbitrary selections rather than just to insertion points.

3.2.3. Names, Keywords, and Keyword/Value Pairs

If a document has many links, the user would benefit from having additional information for each link before choosing which one to follow. Imagine a poem with links leading to the poet's biography, to the poem's location on the poet's timeline, to a document that describes the style of the poem, and to a related poem by an earlier poet. All of the links are marked by identical icons. If a reader who wanted to see only related poems had a way to know which links were relevant, the process of choosing and following links would be more efficient.

The system could allow authors to attach titles, keywords, and arbitrary keyword/value pairs to their links. The title would be an arbitrary character string attached by the author. The user might select simple keywords from an available list or might enter each keyword freely without matching against existing keywords. Keyword/value pairs allow more precise, detailed keyword searching; for example, a professor creating material for a poetry course might add a keyword/value pair *meter=iambic* to one link and *meter=dactylic* to another link. The professor could later ask the system to retrieve links only where *meter=iambic*.

For all links, the system might also have common, predetermined keyword/value pairs to which it assigns values automatically. Possible keywords include the kind of document at the other end of the link, the title of that document, and the creation date of the link. Such information not only could provide additional information on all of the links, but could also be available for querying. A sample query might be, "Show all links created after September, 1986."

Currently, Intermedia incorporates three of these features: titles, keywords, and system-defined keyword/value pairs. The system fully implements titles by providing a default title that the user can edit. A menu command provides access to the title of any link. The functionality is partially developed for both keywords and automatically attached information. Automatically attached data include the date of creation and author's name (since, in Intermedia, more than one author may have access to the same document). Authors can enter an arbitrary number of keywords for any link. Whereas titles and keywords may be edited, the automatic information is fixed.

The use of keyword/value pairs attached to an anchor or link can vary. The issues essentially can be broken into two areas: (1) definition of the search criteria, and (2) feedback of results.

Defining the search criteria is itself a two-part problem: it involves (1) deciding the extent of queries that will be allowed and the domain on which those queries can operate, and (2) determining the appropriate user interface to generate such queries.

Users must be able to specify the *scope* of the queries that use keywords as a criterion. At its simplest, a search for block or link keywords could be limited to the currently active document. At the other extreme, it could extend to all documents in the system. Alternately, it could be limited to all open documents. Or it could be limited to those documents that reside in the currently opened web. Additionally, the user may be able to specify whether the search domain should include keywords attached only to links, only to anchors, or to both.

Document and web owners might want to apply some permanent constraints to the search scope. For example, the web owner might want to dictate that links with the keyword "confidential" are never to be viewed by anyone except him/herself. In the simplest scenario, a set of global web keywords could be established that would always be added to whatever queries an individual makes. In a more complex system, each user might have his or her own "global" keywords that are added to the web owner's keywords, and an owner might be able to apply permanent keyword constraints to each document. On top of all of these owner constraints, a user would want to add his or her own constraints. For most queries, these would likely be per-session temporary query constraints, but for some, the user would want to have the personal constraints applied automatically each time a web or document was opened.

Discussion of the type of user interface needed to specify queries with appropriate scope and content involves a set of extremely complicated issues. In short, these issues include ease vs. power of query language, temporary vs. permanent queries, and graphical vs. textual specification of scope, among others. Many of these issues are the same as those that have troubled designers of data base management systems for years. To that end we will not delve further into these issues in this forum.

Assuming a system designer can bind upon an interface that allows a user to specify the scope and content of a query, the next issue is how to apply the query, and next, how to present the results back to the user. In particular, the query can be applied for one of two purposes: (1) to *filter* the user's domain of links and blocks, or (2) to *collect* the results that satisfy a particular search criterion.

In a multimedia document system, a *filtering* query would typically result in the removal of block and/or link markers in the documents contained in the query's scope. For instance, the query "Show only the link markers for the links created by 'kes' for all the documents in the current web" would result in the "hiding" of all the markers in all documents, other than those that had been created by the user 'kes." In a *collection* query, that same request would result in a list of all link explainers of all links for all documents in the current web that met the criterion. Where a filtering query changes the user's view of the active corpus, a collection query enumerates the results without changing the state of the work.

Hypertext can be used in a clever way to aid the user interface for a collection query. In many data base systems, the result of a collection query is, too often, a list of "hits" quickly typed out on the screen—which must be quickly copied by the user, or, sometimes, captured in a file; users must then manually enter the document name to browse each item discovered. In a current unimplemented Intermedia design, the results of collection queries would themselves be collected in a hypertext document. Each "hit" referencing a block or link would be identified by a text line that includes the document name and block/link explainer. Most important, that line would itself be linked to the block or link destination that was discovered. With this system, a user could simply scroll through the collection list, choose "Follow" for each of the hits that seemed interesting, and travel to exactly the proper place in the corpus, using the same user interface that he or she is familiar with for traversing in the hypermedia.

3.2.4. Enhancing Link Functionality

What kinds of links are appropriate for a particular hypermedia system? An on-line encyclopedia might have only automatically created links from keywords to other articles. There may be no need to have more than one link type. However, in a more general-purpose hypermedia system, users might create links for different purposes. One user might create links only to connect footnotes to their sources, while another user might want to link vocabulary words in a text to their definitions.

With the basic manual link type—a simple tie between two anchors—users can still perform basic linking tasks, such as linking footnotes and dictionary entries. But, by specializing links to do a specific task, the system could make it easier to create certain kinds of links. One such type of link is a *numbered link*. For example, a footnote link type might be represented by a special automatically-numbered icon that appeared in the source document.

Automatic links would be generated by the system rather than the user, based upon the contents of the target documents and the algorithms supplied by the user. For instance, a vocabulary link type might automatically link each word in a document to the appropriate location in a dictionary file; each word in the user's files essentially becomes a source anchor while each entry in the dictionary becomes a destination block pointed to by one or more source anchor words.

Additional link types could provide functionality that is not possible with just a simple tie between two anchors. For example, link destination could depend on such factors as the user's level of expertise, the actual content of the source or destination anchors, or whether the user had previously taken the same path. These *conditional* links would be useful in some educational and training situations.

An *action* link, rather than bringing up a new document, could cause a certain operation or computation to be performed each time it was followed. Essentially, the link is attached to an action formula, like that of a spreadsheet cell, that is evaluated on the follow; it can use the content of the source or destination anchors as values if desired. In a more complex system, the formula could allow references to other links/link computations. If parts of the formula could themselves be references to hypermedia commands, each one of these action formulas can be thought of as a potential system macro.

The first implementation of Intermedia does not provide numbered, automatic, conditional, or action links. Future plans include both footnote and automatic dictionary links as described above.

3.3. Visualizing Connectivity

Additional tools could be useful in helping the user understand the information structure of the linked network. One such tool might be a graphic representation, or map, of a web of links.

The scope of such maps is an important consideration. A *global map* could portray the entire linked structure, similar to a highway map of the United States. However, global maps become difficult to display in a way that is helpful to the user—both in systems without partitioned contexts and in systems with partitioned contexts but with a large number of documents. These "flat" representations often become tangles of lines. Three-dimensional global maps could untangle the lines somewhat to clarify the linking structure. However, users might be confused when viewing such a map.

Alternately, many systems show a global map by picking some document as the root and representing the network of links as a hierarchical tree. If there are cycles in the network, these schemes either duplicate the relevant subnode of the hierarchy (since the maps are typically generated only for what is currently in view, infinite recursion is prevented) or draw lines back to a previously drawn node. In the latter case, a tangle can again result, so many systems have viewing modes that allow users to see the hierarchy both with and without cycles.

In all of these systems, the notion of fish-eye views [Furn86] allows the user to focus in detail on the most important information while seeing, in less detail but in spatial context, much of the information that surrounds the main information.

A subset of a global map could simplify the user's cognitive world model. A *local map* could show those links that emanate from the current document, like an inset of a road map that shows only the roads that lead directly from Providence to other cities. A local map might exist as a static snapshot of the possible links from one anchor or document or as a tracking map that updates itself as the user moves from one document to the next, always keeping the current document as the central focus.

In a system in which links can lead to anchors inside documents, system designers need to determine what level of detail is appropriate to show in a map: a document to all of its related anchors or to only its related documents, or an anchor to related anchors or just to related documents. Global and local maps have different potential levels of detail. A global map could show how anchors are linked together or how documents are connected. Furthermore, a global map could provide both levels of detail and illustrate how both anchors and documents are connected [Figure 19]. With local maps, there are more possibilities. As with global maps, local maps could show links between anchors, links between documents, or a combination showing both levels of detail; but unlike the global maps, the connections would be shown only from the focus, the current document, or the anchor. In addition, because local maps have a focus, they can show the documents that are linked to or from the current anchor, or the anchors that are linked to or from the current document [Figure 20].

Additional map features could include a label on each link and an indication of the direction of each link. If filtering is implemented, the map could represent a subset of the web as processed through the active filters. The user could control these features, or the system could impose them.

Intermedia provides both global and local maps showing document interconnection. Future versions will provide additional map functionality.

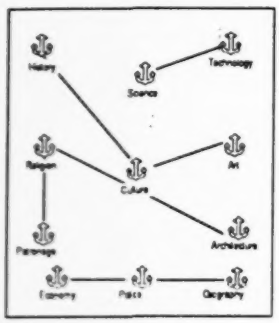


Figure 19a. Global maps might show links between anchors,

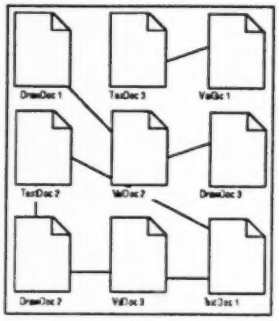


Figure 19b. ...links between documents,

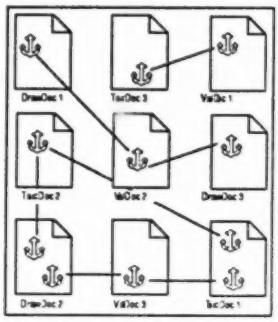


Figure 19c. ...or in both levels of detail.

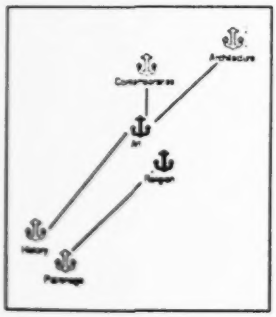


Figure 20a. Local maps might show links between the anchors in the current document to other anchors,

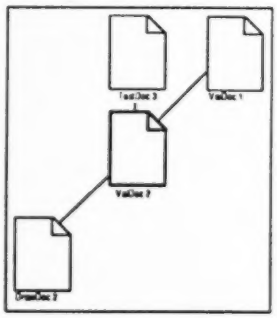


Figure 20b. ...the current document to other documents,

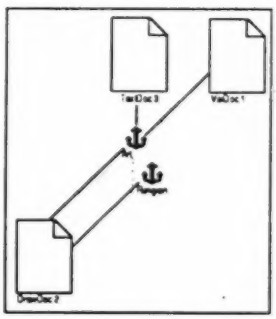


Figure 20c. ...the anchors in the current document to other documents,

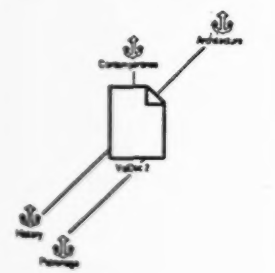


Figure 20d. ...the current document to other anchors,

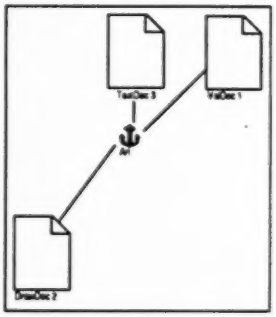


Figure 20e. ...the current anchor to documents,

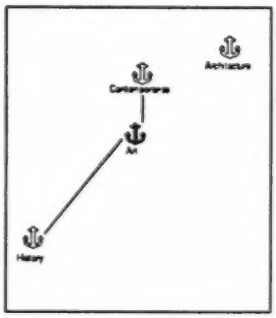


Figure 20f. ...or the current anchor to other anchors.

To be effective for cooperative work, a hypermedia system must provide an authoring as well as a browsing capability. Adding this capability to a hypermedia system increases its complexity. Designers need to determine how the system should respond to editing outside of the confines of anchors, changing the contents of an anchor, and deleting anchor contents.

The effects of editing are influenced by the cognitive model of anchors that the designer decides to present to the user. An anchor can be either a list of distinct objects or a boundary that the user specifies at link creation. The former scheme means that the user can clearly define the anchor, but the system has to store more information per anchor. The latter is more efficient as far as storage is concerned, but the potential for ambiguity is greater. Different media might benefit from different schemes for representing anchors. For example, anchors in a text editor are well represented by all of the text falling between two endpoints. In a structured graphics editor, on the other hand, anchors are better represented as a list of graphics objects. Even with a defined cognitive model, the designer must anticipate and resolve conflicts that can occur during normal editing. The remainder of this section discusses the details of these concerns.

4.1. Editing Content Within Anchors

With a pure boundary scheme, a text editor would represent an anchor made from the string "green eggs and ham" as all of the characters between the "g" in "green" and the "m" in "ham," regardless of what might be changed between these two endpoints. When the user inserts the word "fried," the phrase would read "green fried eggs and ham," and the anchor would now reference the entire phrase. Therefore, by editing, the user could change the anchor contents. In many cases such simultaneous changes of document content and anchor content would be appropriate.

However, the integrity of a link could be at stake if the change is at the boundary of the anchor or if there are substantial changes to the anchor contents. If the user links "eggs and ham" to a recipe, then later inserts "green," so that the phrase now reads "green eggs and ham," the link may no longer be appropriate. Similarly, if the user links "I do not like green eggs and ham" to a biography of Dr. Seuss and then substitutes "scrambled" for "green," the link might not be valid. Instead of automatically updating the link anchor to include the editing changes, the system could eliminate the link on the assumption that any changes to the anchor would invalidate the link. A more flexible system could alert the user that there were changes to an anchor and allow him or her to determine whether the link should remain.

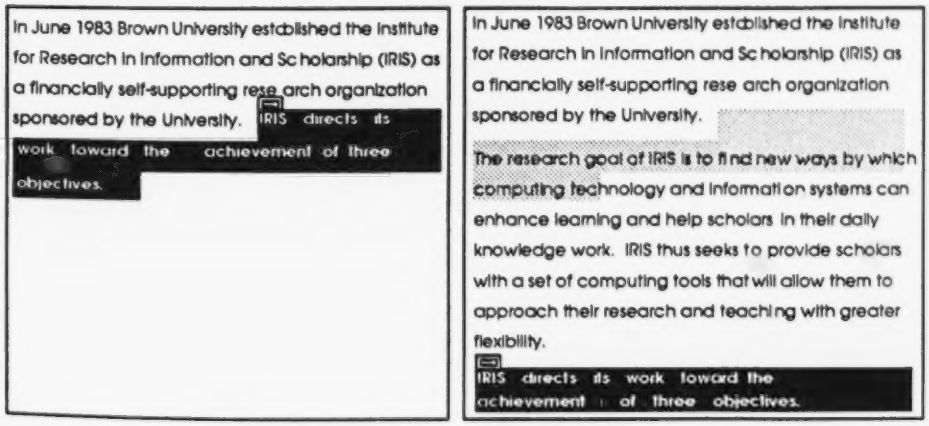
The list scheme may solve some of these ambiguity problems, but it introduces new problems. With the pure list scheme, the user can select two wheels in a picture of a jeep and link them to a description of snow tires. If the user then adds an axle between the wheels, the axle would not automatically become part of the anchor; the system would appear to be correctly anticipating the user's intention. However, this scheme presents new design problems, such as how to let the user add objects to an existing anchor and how the system should respond to changes in any of the anchor's objects. For example, the user might want to add the other two wheels to the anchor. And, if the wheels were replaced by cement blocks, the author would probably not want the link to remain.

In Intermedia, anchors in graphics documents are object lists, and the addition of a graphics object does not automatically change existing anchors. Currently, anchors in text documents are limited to insertion points. Future versions of Intermedia will implement the endpoint paradigm for anchors in text documents.

4.2. Editing Content Outside of Anchors

A system with read/write documents needs to ensure that editing changes outside of anchors do not change what is referenced by the anchors [Figure 21]. This is a problem only in editors of linear data, such as text, that use the boundary scheme. In such editors the insertion of new data "bumps" existing data. Therefore, the system might need to adjust its internal anchor representation to insure that the proper part of the document is presented as an anchor to the user. By contrast, under the list scheme, editing changes outside of an anchor would not affect the identification of the objects to which a link is anchored.

has established a
(b) When the
a new paragraph,
needs to adjust the
the anchor


In June 1983 Brown University estdblished the Institute for Research in Information and Sc holarship (IRIS) as a financially self-supporting rese arch organization sponsored by the University.
IRIS directs its work toward the achievement of three objectives.
In June 1983 Brown University estdblished the Institute for Research in Information and Sc holarship (IRIS) as a financially self-supporting rese arch organization sponsored by the University.
The research goal of IRIS is to find new ways by which computing technology and information systems can enhance learning and help scholars in their daily knowledge work. IRIS thus seeks to provide scholars with a set of computing tools that will allow them to approach their research and teaching with greater flexibility.
IRIS directs its work toward the achievement of three objectives.

4.3. Deleting the Entire Contents of an Anchor

When the user deletes the entire contents of an anchor, should the system also delete the attached links? Although it would be possible to implement such a scheme by leaving the link information attached to an insertion point anchor, it might not be appropriate to maintain a reference to something that no longer exists. For example, if a reference to Dylan Thomas's "A Child's Christmas in Wales" linked to a copy of the poem itself, an author would probably not want to maintain the link if the reference were eliminated. Alternatively, if the user wanted to cut the reference only to replace it with more accurate information, it might be appropriate for the system to attach the link to the new content. There is no way to predict whether the link would still be relevant. A modification to the above alternatives (in which the system arbitrarily maintains or eliminates link information) is to leave the decision to the user on a case-by-case basis.

Intermedia retains the link even when all of the anchor's contents are cut, to avoid being destructive. The user can then explicitly remove the link if necessary. This solution allows the user to control what links are left in each instance and recognizes that actions on content and actions on link information are different.

A Multiuser Environment

If a hypermedia system is to be useful for cooperative work, it must provide ways for multiple authors to read, link to and from, and even edit the same set of documents, while also allowing authors to protect their work from unauthorized access or changes. At its simplest, this functionality could be implemented in a system that gives authors sequential access to documents. However, it would be more useful for cooperative work to have a fully networked environment in which multiple authors would have simultaneous access to documents.

The possibility of multiple users examining and editing documents simultaneously introduces complexity to any system. In hypermedia systems the complexity is increased because the user might not need to edit a document but might want to link to or from the document contents. Interactive linking adds another dimension to the traditional read/write access-rights paradigm. The issues involve control of access to documents and webs, data storage design for both document and link data, contention management, and update schemes for propagating editing changes.

5.1. Access Control

In a system that promotes shared, cooperative work, it is important to control access to document contents and link information. Appropriate access control assures the integrity of information, allows individuals working together to have full access to their shared data, and prevents unauthorized users from viewing or modifying such data.

5.1.1. Document Access

Typically, systems provide two categories of document-access privileges. *Read access* allows the user to view data, while *write access* allows the user to modify data. These categories are still important in the hypermedia world, but they are complicated by the hypermedia notion of *annotate access*. Such a category would be appropriate for systems that encourage users not only to read documents and follow links, but also to create new links as they browse. With annotate access, users could create links without having the right to make changes to the content of the document. Presumably, these access rights are additive, meaning that write access encompasses annotate and read access, and annotate access includes read access.

The annotate privilege could be: (1) limited to writers of documents, (2) assigned to all those with read access, or (3) designated as a separate category. The first case is most appropriate when it is necessary to prevent readers from changing the system in any way, such as in delivery of training materials. The case in which it is appropriate for everyone with read access to make links is in a system that encourages maximum connectivity of information, as in a university course that emphasizes discovery of meaningful relationships among data. The third case provides the flexibility and protection necessary for effective cooperative work.

Intermedia provides all three document-access categories. *Readers* may examine the contents of a document and follow links. *Annotators*, in addition to having readers' rights, can add links and can modify their links, but cannot alter the contents of the document. *Writers* have all of the capabilities of annotators and can also modify the contents of a document.

5.1.2. Web Access

A system with webs, or separate link contexts, could also control access to the webs. The web-access paradigm might follow the document-access categories of read, annotate, and write. Users would need *read access* to navigate the links associated with a web. As in the document-access paradigm, users would need *annotate access* to add links to a web. *Writers* would have all read and annotate rights plus the right to delete or rename the web. To determine if the user could access or change any document and follow or establish any link, the system would need to match the web-access rights against the particular document-access rights of the user. For example, to browse a web, the user would need not only read access to the web but also at least read access to each document that he or she wanted to view.

5.1.3. Assigning Access Rights

After designers specify the categories of access rights, they need to determine how these privileges should be assigned. One possibility is that the creator of the document or web be the only person with rights to modify any contents or links, and all other users have only read access. Another possibility is to designate a document as either private or public; once a document is launched into the public domain, all users would have all rights to it. Alternatively, users could have only "constructive" rights, as opposed to "destructive" rights, allowing them to add information but not to remove any information from the system.

Intermedia designers addressed the problem by designating one *owner* for each document and web and allowing the owner to designate individuals as having read, annotate, or write access to that document and web. Such users may be referred to either individually, by specified lists of users, by groups, or by default as the remainder of the *universe*. When a document opens, its access rights are combined with the rights of the currently open web to create an access-rights *capability* for that document. For example, if the document-access rights allow annotation but the web-access rights allow only read access to the web, the document capability would be set to allow only read access. This allows individual owners to specify the rights they desire for their document, while allowing the owner of the web to set limits for that web.

5.2. Storing Document Content and Anchor Information

Between sessions, anchor information could be stored with document content or separately. Because the anchor information is closely related to the content, it seems natural to store the two kinds of data together. In a system in which only one user at a time can annotate or edit a document, this approach might be appropriate. But if the hypermedia system allows (1) multiple simultaneous annotators, (2) different webs imposed on the same document at the same time, or (3) static documents such as those available on compact disks, the system designer must consider a different approach.

If the system allows multiple simultaneous annotators, it would be unwise to use the one-file system; allowing multiple users to write to the same file simultaneously presents potential concurrency problems that are not handled by most operating systems. Rather, the system should provide a way for multiple individuals to access and update the anchor information without fear of write collisions. Furthermore, in the one-file scheme, the document would have to store anchor information for all webs. Each application would need to filter the link information on a per-web basis.

The approach that we have taken is to store the link data independently from the document contents that they reference. Such data is stored by a data base management system on a per-web basis. When a document opens, the data base is queried for all of the document's link information in the currently opened web; the links are then dynamically added to the document's internal data structure. The data base management system provides concurrency management at the record-locking level over a network of workstations, so that no two individuals can update the same link information at the same time. Updates to the data base are not made at the time of link creation/deletion, but are batched together using a transaction-based scheme and are added to the data base when the web is saved. Because the link information is stored independently from a document, user A can have web A opened on a set of read-only documents, user B can have web B opened on the same set of documents, and both can add or delete links and save their respective webs without modifying the document files at all. Additionally, we anticipate that such a scheme will facilitate adding hypermedia structure to large-scale CD-ROM document sets.

5.3. Contention Management

If the system does not limit the number of users with write access at the document level, it must provide a scheme to manage simultaneous users with equal rights at run time to assure data integrity. Plans for contention management, which are well known for conventional read-write access situations, are complicated by the addition of annotate access to the system. For each document there are eight possible combinations of current status: below, s = single, m = multiple.

writer	annotate	reader
s	s	s
s	s	m
s	m	s
s	m	m
m	s	s
m	s	m
m	m	s
m	m	m

On this table, "s s s" denotes a single user accessing with write, annotate, and read capabilities, and "s s m" denotes a single user accessing with write and annotate capabilities, but multiple-user access for reading.

The hypermedia system designer needs to determine which combinations are acceptable. Some are far more difficult to implement than others. Of the first four, "s s s" and "s s m" are the easiest to build. "s m s" and "s m m" are significantly more difficult to implement than the previous two because of the concurrency problems involved with multiple simultaneous annotations. They require that the access to information in the web stay synchronized while multiple users add and delete links from the web and another user simultaneously makes changes to the document contents. The last four include multiple writer privilege and require complex chalk-passing and synchronization protocols to enable multiple individuals to edit simultaneously.

In Intermedia, when a user opens a web or opens a document from within a web, the system computes a capability for that user for that session for that object. For this computation, the system matches the user's rights to access the object (document or web) against the system's maximum allowed access level (one version of the system, for instance, might not allow multiple simultaneous annotation even if the individual has those rights). Once this capability is computed, the document is opened on a read-only basis if the user has read rights. If not, the user is presented with a dialog that signals the lack of rights, and is unable even to read the document.

If the document is opened on a read-only basis, the user is free to browse through the document and follow links. The first time the user attempts to make a change that affects the contents or the links of the document, the system checks first to see if another individual has locked that document for writing/annotation. If so, the user is told that he or she is unable to make changes to that document currently, but is free to read it and follow links from it. If the document is not locked, the system checks to see if the document capability allows writing/annotating. If not, the user is told that he or she is unable to edit/annotate, and appropriate menu items are disabled. If the document does have the appropriate capability, it is locked in the data base to alert other users who may be trying to annotate or edit it. The user interface is such that individuals attempting simply to read documents and follow links in a corpus that gives them appropriate read rights never see any signs of an existing access-rights scheme. Only individuals annotating or editing the corpus are aware of potential contention conflicts, and this only when there are actual locks or version mismatches that the author wants to be aware of.

5.4. Annotating/Editing Effects on Documents

Notification of updates is another issue to consider in a multiuser environment. There are four possible ways the system could respond to a user's changes: immediate update, immediate notification, passive notification, or no notification. The complexities of these options differ depending on whether the underlying documents are read-only or read/write.

The basis for immediate update is that a user should always have current information. Every new link could be broadcast across the network and would appear immediately whenever that document is open. In a read-only environment this would cause little disruption for the user. The appropriate indication of the new link would appear, and the user could then access the link. However, in a read/write environment this option would pose not only severe implementation problems, but also several conceptual problems. Presumably, if the system were to update links immediately, it should also update the document contents immediately to preserve the integrity of the information. It could be extremely distracting to a reader if the contents of a document were to change automatically because of editing by another user currently accessing the same document.

Immediate notification differs slightly from immediate update in that changes to an open document would result in a message being sent to all current readers of the document, letting them know that someone had changed the document. Users could then request to have their version of the document updated. This would give the reader more control over the changes presented, thereby preventing the problem of unexpected display changes.

Passive notification would inform users, via a facility such as electronic mail, that there had been changes to documents since they last opened them. Such notification would be particularly helpful to users who had made links to these documents; they could then determine whether or not the links should remain considering the changes in the document.

The simplest solution for implementation is not to provide any notification of changes. Open documents would remain as opened, but the next time the document opened it would reflect the changes. These last two alternatives, passive notification and no notification, imply that most link information is not so timely that it needs to be known immediately.

Intermedia currently provides the no-notification scheme as an interim solution. We are investigating ways of allowing users to determine the level of notification they would prefer.

Future Research

There are other issues in hypermedia that we are anxious to explore in future versions of Intermedia. One such issue pertains to sophisticated navigational tools. Such tools could show what links the reader followed to get to the current document, could allow the author to define paths for readers to follow, and could let the user leave a "bookmark" in the corpus to make it easy to "jump" to that place later. We also want to implement filtering so that the user could concentrate on links of particular individual interest in the current document and perhaps throughout an entire web. Furthermore, we are planning on looking into "hot links" that would force an instance of the information at one end of the link into the document at the other end. A hypermedia system that would automatically keep track of versions of documents [Nels81, Deli85] is also an area for future exploration.

Applications of Intermedia

To assess the power and utility of hypermedia, IRIS is conducting a series of experiments at Brown that introduce the Intermedia system into existing courses. As in the English literature course, students in a plant cell biology course are using Intermedia's editors, utilities, and linking functionality to write term papers and to explore a corpus of material about the cell and its processes. Prior to the beginning of the semester, the professor wrote, collected, organized, and linked together primary source material for the corpus, including current research papers, digitized electronmicrographs, diagrams, lecture notes, and three-dimensional models of cells. In writing their papers, students use the same editing tools used by the professor, including links to relevant primary source material in the corpus. The professor annotates student papers at various stages in the writing process, beginning with the outlines. These annotations consist of notes about the content of the paper and links to material that the professor feels the student should read, examine, or explore before committing to a particular assertion or argument. Finally, after the papers are "submitted" for a grade, the students in the course are asked to read and comment on each others' papers. Like the professor's comments, the students' annotations include reactions and criticisms, or point to other student papers or to primary source material. Since the comments become part of the corpus, each student has the opportunity to read any or all of the comments left by others and comment on the comments, thus engaging in an electronic dialogue. At the end of the course, the professor evaluates the quality of each student's research paper as well as the quality of his or her comments.

In the cell biology experiment, the Intermedia system is being used throughout the authoring process, providing facilities to create, organize, reorganize, and cross-reference free-form text and graphics. The experiment also illustrates how a group of authors and editors can use a hypertext or hypermedia system to work collaboratively, using the tools to communicate as well as create.

In the Intermedia system, hypermedia functionality is incorporated into an integrated work environment with consistent, modeless, direct-manipulation applications. Strict adherence to predefined user-interface paradigms throughout the system eases the difficulty of learning a set of different applications. The ability for users to apply what they have already learned from using one application is particularly important in an environment, such as Intermedia, that encourages frequent transitions from application to application.

Problems With Hypermedia Systems

Hypermedia is a powerful concept for multimedia communication. The goal of presenting a "seamless" environment—one in which a reader can become an annotator or an author—presents interesting user-interface problems. The goal of presenting the same seamless environment over a large corpus of documents that can be read, annotated, and edited simultaneously by multiple users over a communications network presents comparably difficult implementation issues. And the goal of presenting a new paradigm for sharing information in such an integrated fashion poses similarly difficult questions of social interaction and the management of such systems. With Intermedia, we have made a first attempt at resolving some of these issues, and we hope that this paper will help us and others as we continue to look at the complicated but fascinating unanswered questions of hypermedia.

Acknowledgements

We wish to thank Helen DeAndrade for designing the figures for this chapter as well as many of the icons, palettes, and dialogs within the Intermedia system.

The development of Intermedia has been the culmination of over a year of intense, coordinated effort by the members of a large team, all of whom equally share the credit for the determination and resolution of the issues explored above. In particular, Charlie Evett developed a large part of the Intermedia architecture, the graphics building block, and the InterDraw application. Ed Grossman, Matt Evett, and Tom Stambaugh developed the text building block and the InterText application. Steve Drucker and Bern Haan developed the Intermedia framework. Page Elmore developed the web data base and the web map capability. Ken Utting added printing capability to the system.

In addition, Mike Braca, Dave Bundy, Dan Stone, and Scott Bates provided the base operating system, the data base, and the operating system extensions that enabled Intermedia to run. Brian Chapin and Sam Fulcomer configured the systems countless times. Eric Wolf developed the program for decompressing scanned images. George Landow, David Cody, Glenn Everett, Rob Sullivan, and Suzanne Keen Morley produced the enormous corpus *Context 32: A Web of English Literature* and heroically subjected themselves to the ordeal of teaching an English literature course with the system in its infancy. Andy van Dam, Marty Michel, Bill Shipp, and Don Wolfe provided advice and support throughout the project. Kate Archambault translated much gibberish into English.

This work was sponsored in part through a grant by the Annenberg/CPB Project and a joint study contract with International Business Machines Corporation. In addition, we greatly acknowledge Larry Tesler of Apple Computer, Inc., Jeff Singer and Stan Fleischman of Cadmus Computer, Inc., and Mark Nodine of Bolt Beranek and Newman for their assistance in making available key software that made Intermedia viable.

References

[Appl83] Apple Computer, Inc. *Lisa Office System*. Cupertino, CA, 1983.

[Appl84a] Apple Computer, Inc. *MacWrite*. Cupertino, CA, 1984.

[Appl84b] Apple Computer, Inc. *MacDraw*. Cupertino, CA, 1984.

[Appl85] Apple Computer, Inc. *Inside Macintosh*. vol. 1, 2, and 3. Addison-Wesley Publishing Company, Inc., Reading, MA, 1985.

[Back82] D. Backer and S. Gano. "Dynamically Alterable Videodisk Displays." *Proc. Graphics Interface 82*. Toronto, May 17–21, 1982.

[Bend84] W. Bender. "Imaging and Interactivity." *Fifteenth Joint Conference on Image Technology*. Tokyo, November 26, 1984.

[Deli85] N. Delisle and M. Schwartz. "Neptune: A Hypertext System for CAD Applications." Tektronix Laboratories Technical Report no. CR–85–50. Beaverton, OR, 1985.

[Enge68] D. Engelbart and W. English. "A Research Center for Augmenting Human Intellect." *Proceedings of FJCC*, vol. 33, no. 1. AFIPS Press, Montvale, NJ, Fall 1968, pp. 395–410.

[Fein82] S. Feiner, S. Nagy, and A. van Dam. "An Experimental System for Creating and Presenting Interactive Graphical Documents." *Trans. Graphics*, vol. 1, no. 1, 1982.

[Furn86] G.W. Furnas. "Generalized Fisheye Views." *Proc. CHI '86* Boston, MA, April 13–17, 1986, pp. 16–23.

[Garr86] L. Garrett and K. Smith. "Building a Timeline Editor from Prefab Parts: The Architecture of an Object-Oriented Editor." *Proceedings of OOPSLA* .Portland, OR, September 1986.

[Gold84] A. Goldberg *Smalltalk-80: The Interactive Programming Environment* .Addison-Wesley Publishing Company, Inc. Reading, MA, 1984.

[Haan86] B. Haan, S. Drucker, and N. Yankelovich. "An Object-Oriented Approach to Developing Consistent, Integrated Applications." IRIS Technical Report.

[Hala85] F. Halasz and R. Trigg. *Notecards*. Xerox Palo Alto Research Center, Palo Alto, CA, 1985.

[Inte86] Interleaf. *Workstation Publishing Software: User's Guide*. Ten Canal Park, Cambridge, MA, 1986.

[Know86] KnowledgeSet Corporation *Laser Facts* Monterey, CA, 1986.

[Land86] G. Landow, D. Cody, G. Everett, and R. Sullivan *Context 32* Brown University, Providence, RI, 1986.

[Meyr85] N. Meyrowitz, et al "The Intermedia System: Requirements" IRIS Report, Institute for Research in Information and Scholarship, Providence, RI, September, 1985.

[Meyr86] N. Meyrowitz "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework" *OOPSLA '86 Proceedings* Portland, Oregon, September, 1986.

[Nels81] T. Nelson *Literary Machines* Swarthmore, PA, 1981. Available from author.

[Shne86] B. Shneiderman and J. Morariu *The Interactive Encyclopedia System (TIES)* University of Maryland, College Park, MD, 1986.

[Smit82] D.C. Smith, C. Irby, R. Kimball, and B. Verplank "Designing the Star User Interface" Byte Publications Inc., April 1982.

[Tesl85] L. Tesler "An Introduction to MacApp 0.1" Apple Computer, Inc., Cupertino, CA, February 14, 1985.

[Weye85] S. Weyer and A. Borning "A Prototype Electronic Encyclopedia" *ACM Trans. Office Information Systems*, vol. 3, no. 1 January 1985.

[Xero82] Xerox Corporation *8010 Star Information System Reference Guide* Dallas, TX, 1982.

[Yank85] N. Yankelovich, N. Meyrowitz, and A. van Dam "Reading and Writing the Electronic Book" *IEEE Computer*, vol. 18, no. 10 October 1985, pp. 15–30.

[Yank86] N. Yankelovich"Intermedia: A System for Linking Multimedia Documents" IRIS Technical Report.

[Yank87] N. Yankelovich, D.Cody, and G. Landow "The Creation of a Hypermedia Corpus for English Literature" In press*SIGCUE Bulletin.*

Norman Meyrowitz

Norman Meyrowitz, manager of IRIS's Scholar's Workstation Group, graduated from Brown University in 1981 with a degree in computer science. His major research interests are in the areas of text-processing, user-interface design, and object-oriented programming. He was the senior author, with Andries van Dam, of the most up-to-date and widely referenced article on computer-based text editing, "Interactive Editing Systems." Under a contract with IBM, Meyrowitz has completed research in integrated page-makeup systems and object-oriented programming under Smalltalk-80. He has authored over a dozen technical publications and several major IRIS proposals, including one in 1984 to the Annenberg/CPB Project, which granted $1.5 million to IRIS's Educational Software Project. In 1980, Meyrowitz designed one of the first window systems based on UNIX®.

At IRIS, Meyrowitz manages a group of 12 charged with the design and development of end-user software, and provides technical management for the Intermedia, Educational Software, and Signifier Projects. He also serves as Brown's technical liaison with Apple Computer, Inc. and several other vendors.